

**ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ БІЗНЕСУ ТА СУЧАСНИХ
ТЕХНОЛОГІЙ
ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ**

Допускається до захисту
Завідувач кафедри _____ О.О. Ємець
(підпис)
«_____» _____ 2021 р.

**ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОЇ РОБОТИ
на тему
ТРЕНАЖЕР З ТЕМИ «СОРТУВАННЯ ВКЛЮЧЕННЯМИ З ЛІНІЙНИМ ТА БІНАРНИМ
ПОШУКОМ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «АЛГОРИТМИ ТА
СТРУКТУРИ ДАНИХ» ТА РОЗРОБКА ЙОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Олефіренко Валерій Васильович
_____ «___» _____ 2021 р.
(підпис)
Науковий керівник канд. фіз.-мат. наук, доц. Ємець Олександра Олегівна
_____ «___» _____ 2021 р.
(підпис)

ПОЛТАВА 2021 р.

**ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»**

ЗАТВЕРДЖУЮ

Завідувач кафедри _____ О.О. Ємець
(підпис)

«__» _____ 20__ р.

**ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ ДИПЛОМНОЇ РОБОТИ**

Студент(ка) спеціальності 122 «Комп'ютерні науки та інформаційні технології»

Прізвище, ім'я, по батькові Олефіренко Валерій Васильович.

1. Тема «Тренажер з теми «Сортування включеннями з лінійним та бінарним пошуком» дистанційного навчального курсу «Алгоритми та структури даних» та розробка його програмного забезпечення»

затверджена наказом ректора № 115-Н від «_1_» вересня 2020 р.

Термін подання студентом дипломної роботи _____ «__» _____ 20__ р.

2. Вихідні дані до дипломної роботи _____

3. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Вступ.

1. Постанова задачі.

2. Інформаційний огляд.

2.1. Алгоритм сортування.

2.2. Огляд аналогічних розробок.

2.3. Позитивні та негативні аспекти розглянутих робіт

3. Теоретична частина.

3.1. Огляд матеріалу за темою роботи

3.2. Алгоритмізація задачі за темою роботи

3.3. Блок-схема алгоритму

3.4. Обґрунтування вибору програмних засобів для реалізації завдання роботи.

4. Практична частина.

4.1. Опис процесу програмної реалізації.

4.2. Інструкція по роботі з програмою.

4.3. Перевірка валідності.

Висновки.

4. Перелік графічного матеріалу (з точним визначенням кількості блок-схем, іншого графічного матеріалу) _____

5. Консультанти розділів дипломної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

6. Календарний графік виконання дипломної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій та стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел бібліотек та інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доробка (за необхідністю), рецензування		

Дата видачі завдання «__» _____ 20__ р.

Студент(ка) _____
(підпис)

Науковий керівник _____
(підпис) (науковий ступінь, вчене звання,
ініціали та прізвище)

Результати захисту дипломної роботи

Дипломна робота оцінена на _____
(балів, оцінка за національною
шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «__» _____ 20__ р.

Секретар ЕК _____
(підпис)
(ініціали та прізвище)

РЕФЕРАТ

Записка: 61 с., 58 рис., 10 джерел.

Предмет розробки – тренажер з теми «Сортування включеннями з лінійним та бінарним пошуком».

Мета роботи – створити тренажер з теми «Сортування включеннями з лінійним та бінарним пошуком».

Методи розробки – мова програмування C#, середовище програмування Visual Studio.

Створено алгоритм для тренажеру «Сортування включеннями з лінійним та бінарним пошуком». Нарисовано блок-схему алгоритму. Створено програмну реалізацію тренажеру з теми «Сортування включеннями з лінійним та бінарним пошуком».

Ключові слова: СОРТУВАННЯ ВКЛЮЧЕННЯМИ, ЛІНІЙНЕ СОРТУВАННЯ, БІНАРНЕ СОРТУВАННЯ, ТРЕНАЖЕР.

ВСТУП	4
РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ.....	6
РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД.....	7
2.1. Алгоритм сортування	7
2.2. Огляд аналогічних розробок.....	9
2.3. Позитивні та негативні аспекти розглянутих робіт	17
РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА	19
3.1. Огляд матеріалу за темою роботи.....	19
3.2. Алгоритмізація задачі за темою роботи	34
3.3. Блок-схема алгоритму	36
3.4. Обґрунтування вибору програмних засобів для реалізації завдання роботи	37
РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА	39
4.1. Опис процесу програмної реалізації.....	39
4.2. Інструкція по роботі з програмою.....	48
4.3. Перевірка валідності.....	55
ВИСНОВКИ.....	59
СПИСОК ЛІТЕРАТУРИ.....	60

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ,
ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**

Умовні позначення, символи, одиниці, скорочення, терміни	Пояснення умовних позначень, символів, одиниць, скорочень, термінів
$C\#$	Об'єктно-орієнтована мова програмування
a_n	Масив чисел.
n	Кількість кроків.

ВСТУП

Оскільки останнім часом в зв'язку з пандемією дистанційна освіта стала одним із головних напрямів розвитку та надання освіти тому на даний момент часу створюють велику кількість різних методів надання інформації та для навчання учнів. Застосування в навчанні комп'ютерів значно спрощує надання інформації учням які не мають змоги відвідувати навчальні заклади.

У зв'язку з цим, в останній час масово розробляють різні методи навчання на відстані, які допомагають викладачеві краще надавати інформацію для студентів.

Отже тренажер з теми «Сортування включеннями з лінійним та бінарним пошуком» буде досить корисним в даний момент часу.

Мета курсового проекту – розроблення елементів тренажеру з теми «Сортування включеннями з лінійним та бінарним пошуком» дистанційного навчального курсу «Алгоритми та структури даних» та розробка його програмного забезпечення.

Об'єктом розробки є розробка програмного забезпечення та реалізація елементів тренажеру.

Предмет розробки — програмне реалізація тренажеру з теми «Сортування включеннями з лінійним та бінарним пошуком».

Перелік використаних методів:

- вивчення теоретичного матеріалу по різновидам тренажерів,
- вивчення теоретичного матеріалу з обраної теми,
- програмна реалізація тренажеру.

Курсовий проект складається з трьох розділів. У першому розділі розглянуто постановку задачі. У другому розділі описано значення та приклади алгоритмічно нерозв'язних проблем, значення навчальних програм для різних дисциплін. У третьому розділі представлено огляд матеріалу за темою роботи і алгоритмізацію задачі.

Обсяг пояснювальної записки: 61 стор., в т.ч. основна частина - 57 стор., джерела - 10 назв.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ

Основним завданням дипломної роботи є розробка тренажеру, який навчає темі «Сортування включеннями з лінійним та бінарним пошуком» дистанційного курсу «Алгоритми та структури даних».

Основні завдання роботи:

- описати постановку задачі;
- огляд аналогічних робіт по вибраній темі;
- розробити алгоритм тренажеру та блоксхему;
- опис мови програмування та технології, що були використані при розробці програми;
- описати процес реалізації тренажеру;
- протестувати тренажер на різних етапах;
- описати процес тестування;
- описати необхідну користувачу інструкцію.

Тренажер має відповідати дистанційному курсу «Алгоритми та структури даних» Полтавського університету економіки і торгівлі.

Програму передати на впровадження у відповідний дистанційний курс Полтавського університету економіки і торгівлі.

Програма повинна в разі помилки вказувати правильну відповідь.

РОЗДІЛ 2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Алгоритм сортування

За весь люди придумали безліч алгоритмів сортування [4] — це алгоритм, що розв'язує задачу сортування, тобто здійснює впорядкування лінійного списку (масиву) елементів.

Термін сортування (англ. *sorting*) означає розділення елементів за певними ознаками (сортами) і не дуже точно описує поставлене завдання. Точнішою була б назва впорядкування (англ. *ordering*), але через переважаність слова «порядок» (англ. *order*) різними значеннями, в цьому завданні ним не скористалися.

Для алгоритму сортування (як і для будь-якого іншого сучасного алгоритму) основними характеристиками є:

- Час, необхідний на впорядкування n -елементного масиву. Для значної кількості алгоритмів середній і найгірший час впорядкування n -елементного масиву є $O(n^2)$ це пов'язано з тим, що в них передбачені перестановки елементів, що стоять поряд (різниця між індексами елементів не перевищує деякого заданого числа). Такі алгоритми зазвичай є стабільними, хоча і не ефективними для великих масивів. Інший клас алгоритмів здійснює впорядкування за час $O(n \log n)$. В цих алгоритмах використовується можливість обміну елементів, що знаходяться на будь-якій відстані один від одного.
- Необхідність додаткової пам'яті для сортування. Зазвичай необхідно $O(1)$ пам'яті.
- **Стабільність** (англ. *Stability*) — стабільне сортування не змінює взаємного розташування елементів з однаковими ключами.

Теорема про найкращий час сортування: якщо алгоритм сортування в своїй роботі спирається тільки на операції порівняння двох об'єктів (\leq) і не враховує жодної додаткової інформації про елементи, то він не може впорядкувати масив елементів швидше ніж за $O(n \log n)$ в найгіршому випадку.

За час $O(n^2)$:

- Сортування вибором;
- Сортування вставкою;
- Сортування обміном (сортування бульбашкою);
- Сортування методом бінарної вставки.

За час $O(n \log n)$:

- Плавне сортування;
- Пірамідальне сортування;
- Швидке сортування;
- Сортування злиттям.

За час $O(n)$: з використанням додаткової інформації про елементи

- Сортування підрахунком;
- Сортування за розрядами;
- Сортування комірками.

За час $O(n \log^2 n)$:

- Сортування злиттям модифіковане;
- Сортування Шелла.

За час $O(n!)$:

- Сортування перестановкою;
- Випадкове сортування.

2.2. Огляд аналогічних розробок

Розглянемо вже існуючі тренажери дистанційного курсу «Алгоритми і структури даних».

Тренажер «Рекурсивне породження представлень» (рис. 2.1-2.7) був створений у 2020 р. Шульга Іллею[8]. Тренажер містить як практичні так і теоретичні питання.

На головному вікні програми знаходиться кнопка переходу до тренінгу та безпосередньо інформація про розробника(рис. 2.1).



Рисунок 2.1 – Головне вікно

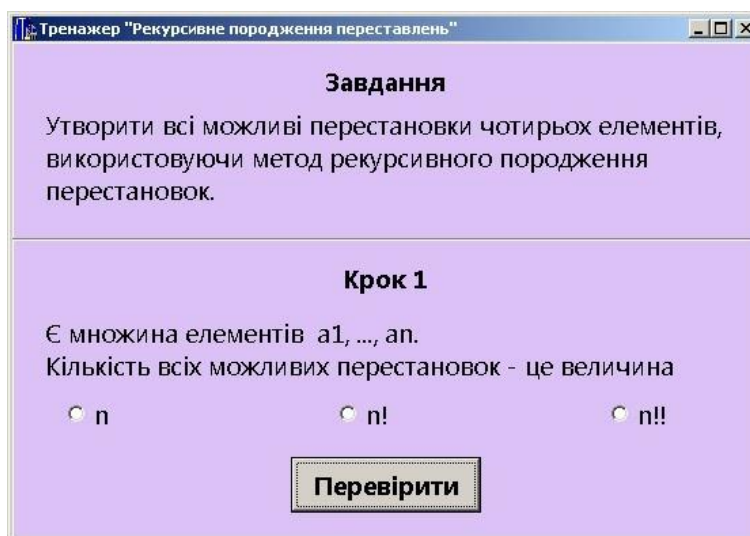
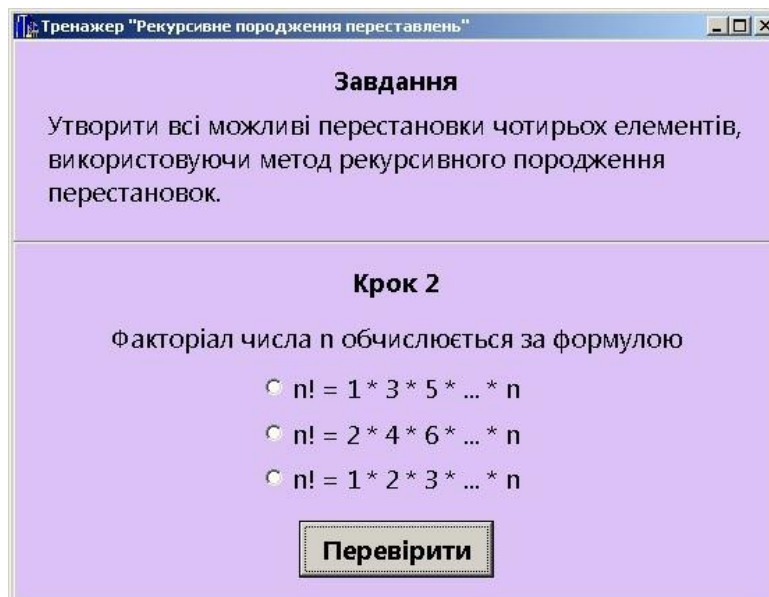


Рисунок 2.2 – Питання №1



Завдання

Утворити всі можливі перестановки чотирьох елементів, використовуючи метод рекурсивного породження перестановок.

Крок 2

Факторіал числа n обчислюється за формулою

☐ $n! = 1 * 3 * 5 * \dots * n$

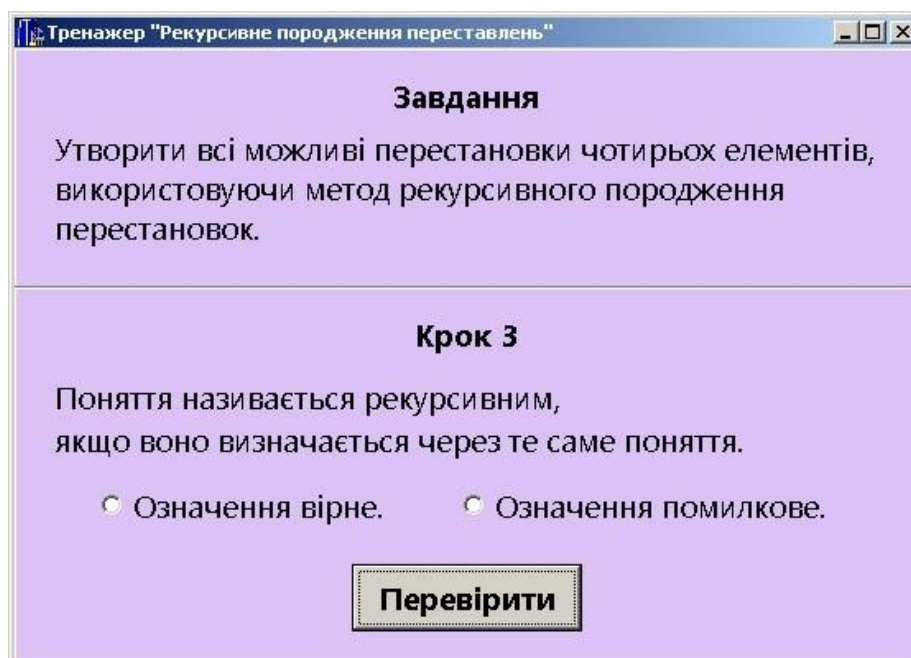
☐ $n! = 2 * 4 * 6 * \dots * n$

☐ $n! = 1 * 2 * 3 * \dots * n$

Перевірити

Рисунок 2.3 – Питання №2

Тренажер містить різні питання за темою роботи, відповіді відбуваються в тестовому форматі або в практичному (рис 2.6).



Завдання

Утворити всі можливі перестановки чотирьох елементів, використовуючи метод рекурсивного породження перестановок.

Крок 3

Поняття називається рекурсивним, якщо воно визначається через те саме поняття.

☐ Означення вірне. ☐ Означення помилкове.

Перевірити

Рисунок 2.4 – Питання №3

Тренажер "Рекурсивне породження переставлень"

Завдання

Утворити всі можливі перестановки чотирьох елементів, використовуючи метод рекурсивного породження перестановок.

Крок 5

Алгоритм рекурсивного породження перестановок:

- 1) Для $n = 1$ - єдина перестановка: 1.
- 2) Нехай на множині з $n - 1$ елементів вже побудовані всі можливі перестановки, P_1, P_2, \dots, P_{n-1} , що задовольняють цій умові. Будемо розширювати кожен із цих перестановок P_i , вставляючи елемент n на кожне з можливих місць справа наліво, якщо i непарне, і зліва направо, якщо i парне.

☐ Алгоритм вірний.
 ☐ Алгоритм з помилками.

Перевірити

Рисунок 2.5 – Питання №5

Тренажер "Рекурсивне породження переставлень"

Завдання

Утворити всі можливі перестановки чотирьох елементів, використовуючи метод рекурсивного породження перестановок.

Крок 6

$n = 1$.
Кількість перестановок з n елементів - це

$n! = 1! =$

Перевірити

Рисунок 2.6 – Питання №6

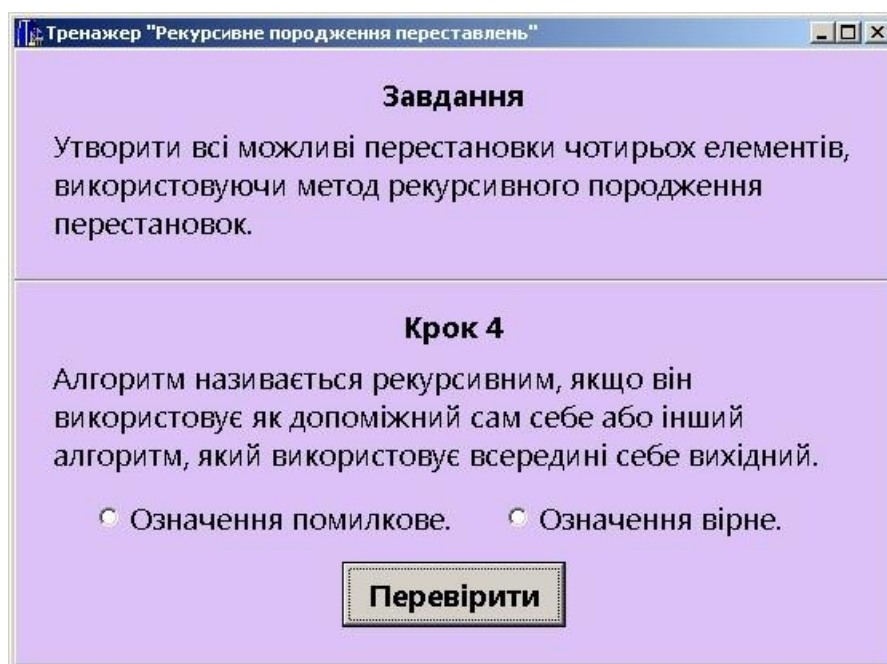


Рисунок 2.7 – Питання №5

Другий тренажер «Лексикографічне породження переставлень» (рис. 2.8-2.12) розроблений у 2019 р. Цебою Миколою.



Рисунок 2.8 – Головне вікно

Лексикографічне породження переставлень

Питання 4

Вкажіть відповіді:

$2! = 1 * 2 =$

$3! = 1 * 2 * 3 =$

$4! = 1 * 2 * 3 * 4 =$

$5! = 1 * 2 * 3 * 4 * 5 =$

Рисунок 2.9 – Питання №4

Лексикографічне породження переставлень

Питання 18

Початкова конфігурація: 1, 2, 3, ..., n.
Кінцева конфігурація: n, n-1, ..., 2, 1.
Умова закінчення: при розгляді поточної конфігурації справа наліво не трапився жоден елемент, менший за попередній.
 $p = \{p_1, p_2, \dots, p_n\}$ - поточна конфігурація.

Алгоритм перетворення конфігурації p на наступну:

- 1) переглядаємо p справа наліво, поки не трапиться елемент $p_i < p_{i+1}$. Фіксуємо його номер i;
- 2) переглядаємо p справа наліво у пошуках першого справа (тобто найменшого) елемента $p_j > p_i$. Фіксуємо його номер j;
- 3) міняємо місцями p_j і p_i , тобто $p_3 = 2$ і $p_2 = 1$
Отримали:
- 4) інвертуємо (записуємо у зворотному порядку) відрізок послідовності p_{i+1}, \dots, p_n
Тобто інвертуємо відрізок p_3
Отримали:

Питання 18

$p = \{3, 1, 2\}$ - поточна конфігурація.
Перейдіть до наступної.

1) переглядаємо p справа наліво, поки не трапиться елемент $p_i < p_{i+1}$
Отже, порівнюємо пару чисел 1 та 2: 1 2
Фіксуємо номер i: i =

2) переглядаємо p справа наліво у пошуках справа (тобто найменшого) елемента $p_j > p_i$
Отже, порівнюємо пару чисел 2 та $p_2 = 1$: 2 1
Фіксуємо номер j: j =

3) міняємо місцями p_j і p_i , тобто $p_3 = 2$ і $p_2 = 1$
Отримали:

4) інвертуємо (записуємо у зворотному порядку) відрізок послідовності p_{i+1}, \dots, p_n
Тобто інвертуємо відрізок p_3
Отримали:

Рисунок 2.10 – Питання №18

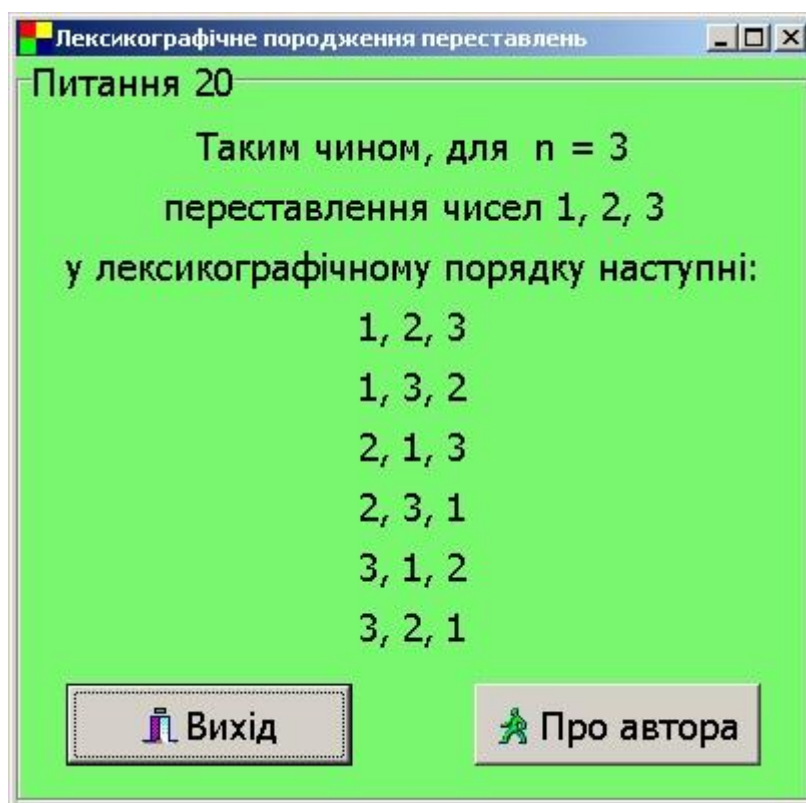


Рисунок 2.11 – Питання №20

Тренажер містить інформацію про автора який створив програму.

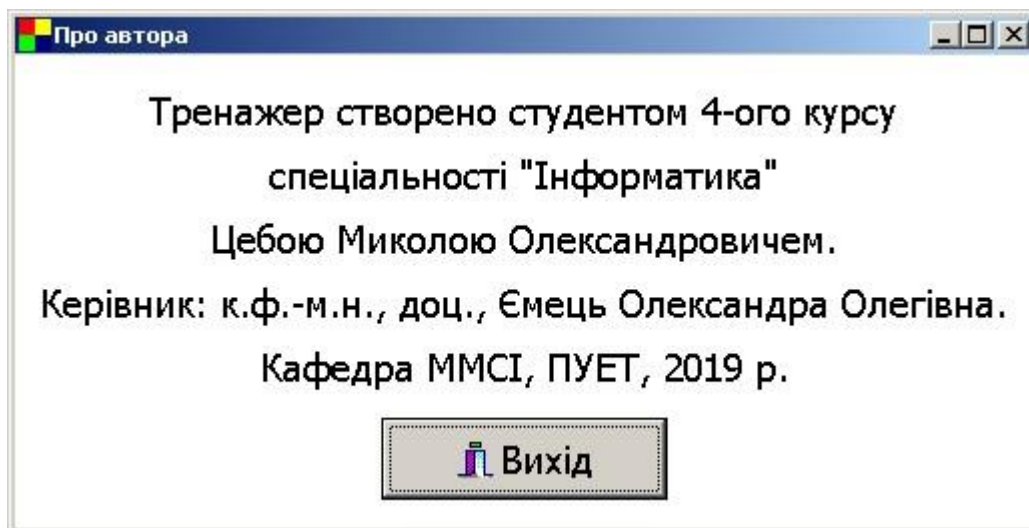


Рисунок 2.12 – Інформація про розробника

Третій тренажер «Пірамідальне сортування» (рис. 2.13-2.17) був створений у 2019 р. Самборською Ксенією.

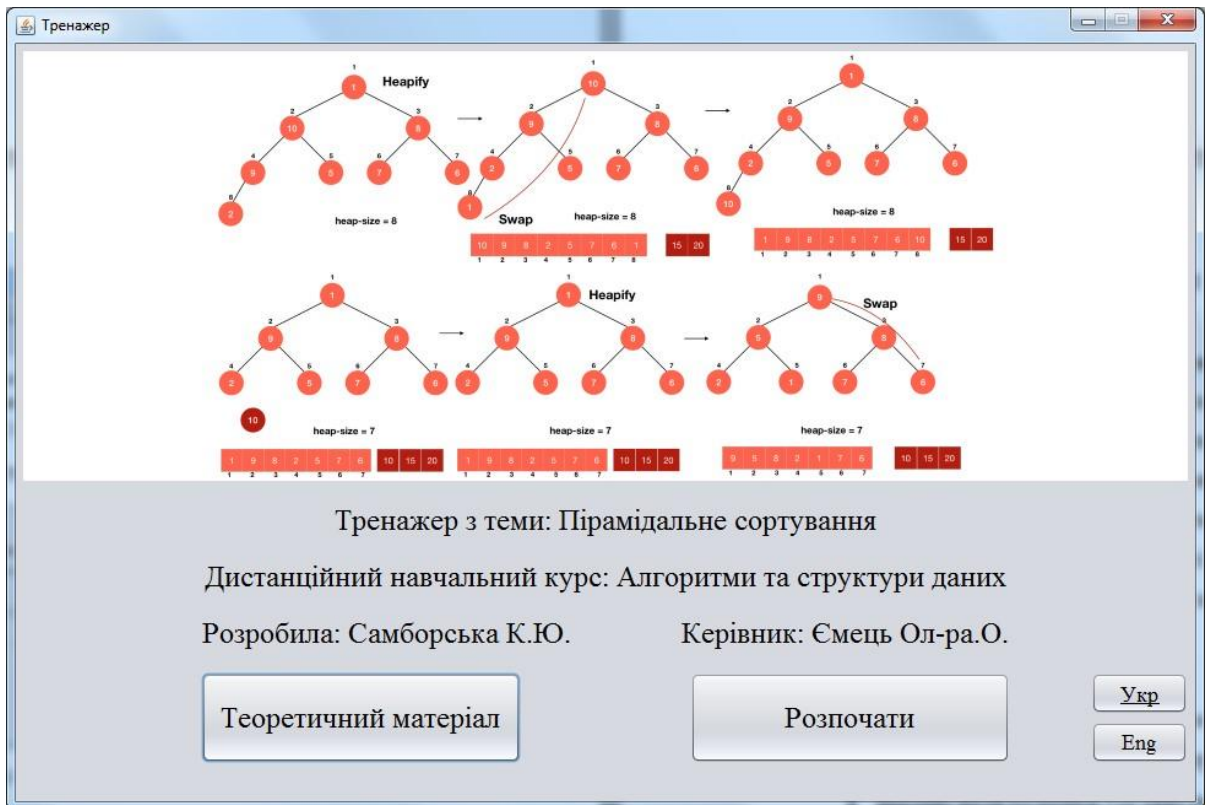


Рисунок 2.13 – Головна сторінка

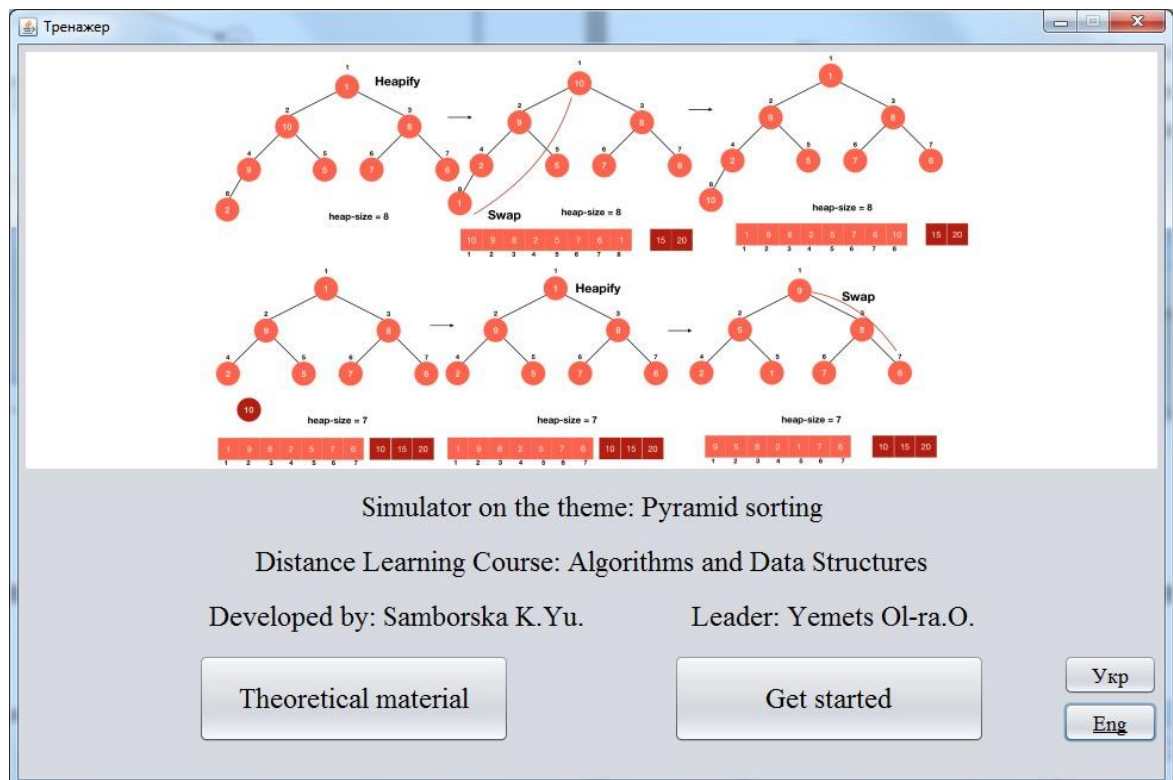


Рисунок 2.14 – Головна сторінка на Англійській мові

Тренажер

Назад

Метод пірамідального сортування поєднує переваги використання деревоподібних структур із сортування на місці, тобто всередині вихідної послідовності. Він належить до групи вдосконалених методів вибору з ефективністю сортування порядку $n \log_2 n$.

Пірамідою називається послідовність елементів h_1, \dots, h_r , така, що $h_i \leq h_{2i}$, $h_i \leq h_{2i+1}$ при $i = 1, \dots, r/2$. Наприклад, послідовність $h_1=3, h_2=10, h_3=7, h_4=12, h_5=10, h_6=8, h_7=12$ – це піраміда.

Піраміду зручно зображати у вигляді дерева (рис. 3.1).

Рисунок 3.1 – Піраміда у вигляді дерева

У вершині піраміди (елемент h_1) завжди міститься її мінімальний елемент. Щоб додати елемент до піраміди, не порушуючи її умов, використовують метод, що називається просіванням: елемент h_i , який додається, послідовно порівнюється з елементами h_{2i} і h_{2i+1} й у випадку порушення умови піраміди обмінюється значеннями з меншим із них.

Метод пірамідального сортування включає два основні етапи (сортуємо за спаданням)

Рисунок 2.15 – Теоретичні відомості по темі

Тренажер

770	504	901	834	488	671	993	948	223	909	902	189	677	172	929	348	326	949	373	639	345	562	913	659	129	247	567	594	190	769	871
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Крок 1. Беремо елемент h_{14} та порівнюємо з h_{29} та h_{30} . Чи порушується правило піраміди?

☐ Ні, не порушується.

☐ Так, порушується. Міняємо місцями h_{14} та h_{29} .

☐ Так, порушується. Міняємо місцями h_{14} та h_{30} .

Вибрати

Рисунок 2.16 – Крок тренажера

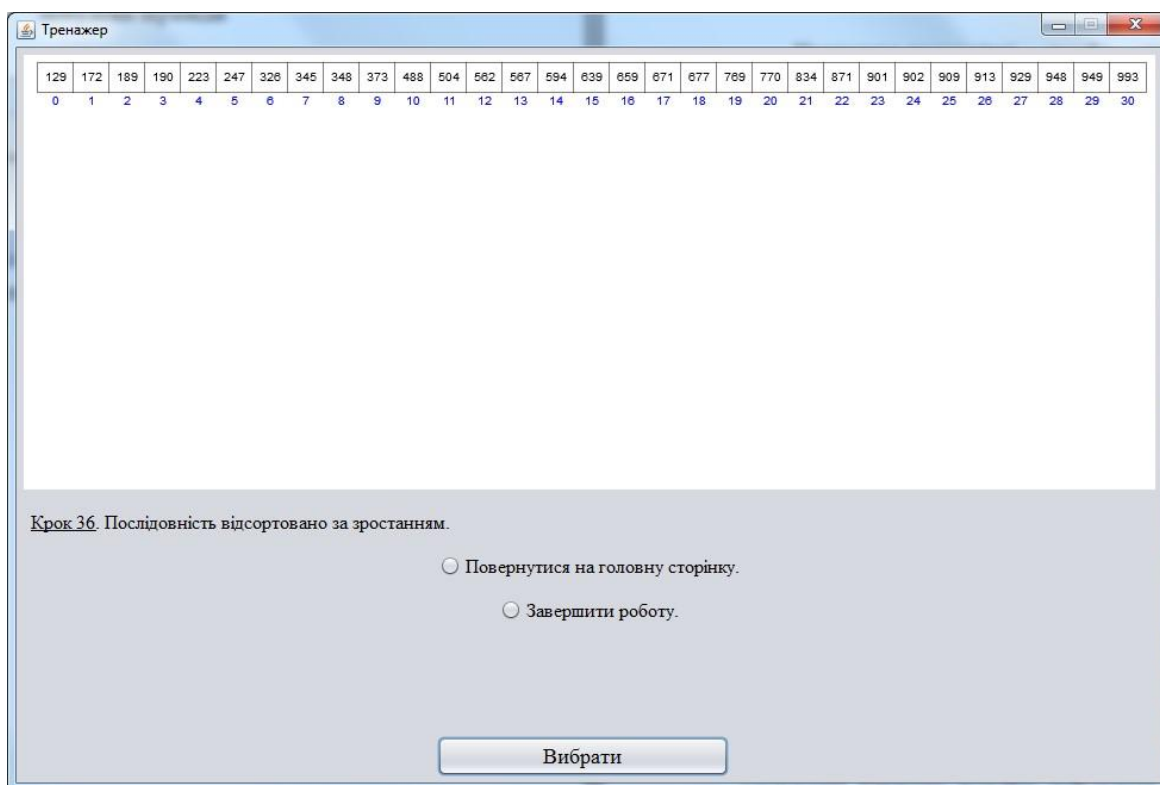


Рисунок 2.17 – Вікно завершення

2.3. Позитивні та негативні аспекти розглянутих робіт

Перший тренажер «Рекурсивне породження представлень».

Позитивні аспекти розглянутої розробки:

- повно розрита тема;
- використано термінологію та позначки з дистанційного курсу;
- вказана інформація про автора тренажера;
- у дизайні використано різні кольори кольори;
- добре сформовані питання.

Негативні аспекти розглянутої розробки:

- значних недоліків не виявлено.

Другий тренажер «Лексикографічне породження переставлень».

Позитивні аспекти розглянутої розробки:

- повна розритої теми;
- термінологію та умовні позначки відповідають дистанційного курсу;

- приємний дизайн тренажеру;
- присутня інформація про розробника тренажера.

Негативні аспекти розглянутої розробки:

- значних недоліків не виявлено.

Третій тренажер «Пірамідальне сортування».

Позитивні аспекти розглянутої розробки:

- вибір між двома мовами;
- присутні теоретичні відомості;
- присутня інформація про розробника тренажера.

Негативні аспекти розглянутої розробки:

- значних недоліків не виявлено.

Необхідність та актуальність теми

Переглянувши тренажери я прийшов до висновку що тренажери допомагають студентам в вивченні навчального матеріалу та покращує його рівень.

РОЗДІЛ 3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Огляд матеріалу за темою роботи

Сортування включенням або сортування вставлянням [3] — простий алгоритм сортування на основі порівнянь. На великих масивах є значно менш ефективним за такі алгоритми, як швидке сортування, пірамідальне сортування та сортування злиттям. Однак, має цілу низку переваг:

- простота у реалізації
- ефективний (зазвичай) на маленьких масивах
- ефективний при сортуванні масивів, дані в яких вже непогано відсортовані
- на практиці ефективніший за більшість інших квадратичних алгоритмів
- є стабільним алгоритмом

Наприклад, більшість людей при сортуванні колоди гральних карт, використовують метод, схожий на алгоритм сортування включенням.

Означення 1. За цим методом уся послідовність розбивається на вже відсортовану частину і ще не відсортовану. Спочатку відсортована частина складається з одного числа a_1 , а інші числа a_1, \dots, a_n утворюють невідсортовану частину. На кожному кроці перше число з невідсортованої частини вставляється (включається, звідси і назва методу) на відповідне місце у відсортованій частині. Таким чином, кількість чисел у відсортованій частині після кожного кроку збільшується на одиницю. Процес завершується за $n - 1$ крок. Проілюструємо метод на прикладі тих самих чисел, що використовували в сортуванні вибором: **5, 28, -1, 46, 33** (рис. 3.1).

Основною підзадачею цього методу є пошук місця для вставки числа в уже відсортовану частину. Вона розв'язується на кожному кроці методу.

Вхідними даними для неї додатково до n ; a_1, \dots, a_n є номер (позначимо його k) першого, ще не переглянутого елемента (тобто довжина вже відсортованої частини на цьому кроці дорівнює $k - 1$) (рис. 3.2).

Результат – номер, позначимо його j ($1 \leq j \leq k$): число, що вставляється (a_k), має стати j -м елементом відсортованої частини.

Назвемо алгоритм цієї підзадачі *Пошук (k)*.

Наступна підзадача – власне вставка елемента a_k на задане місце. Вхідним для неї є номер j (вставити на j -те місце), а результатом – упорядкована послідовність завдовжки k . Назвемо алгоритм розв'язування цієї *Вставка (j)*. З використанням цих алгоритмів загальний алгоритм сортування методом включень можна записати у такому вигляді, як зображено на рис. 3.3.

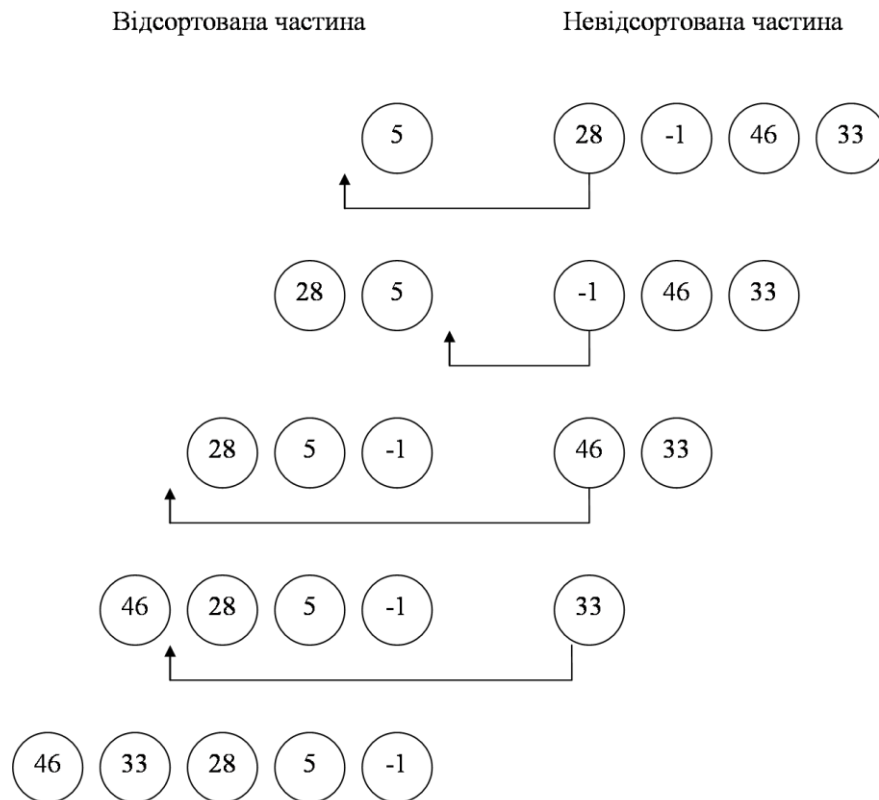


Рисунок 3.1 - Ілюстрація сортування методом включень



Рисунок 3.2

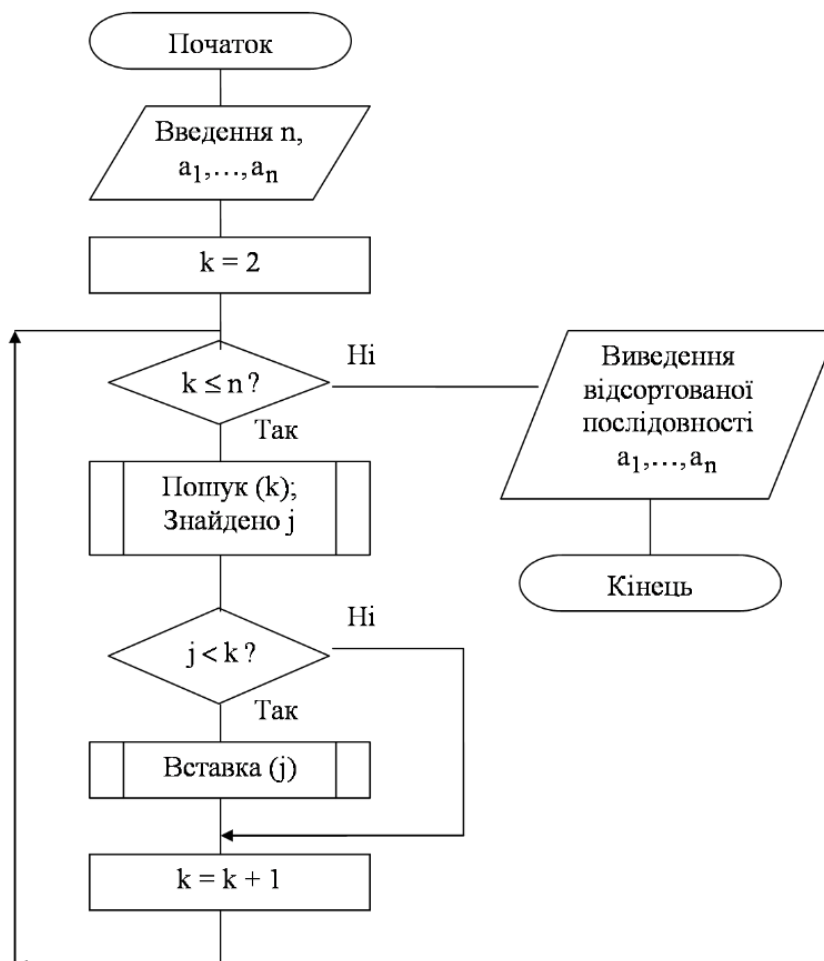


Рисунок 3.3 - Загальний алгоритм сортування методом включень

Тепер можна докладніше розглянути задачі пошуку місця для вставки і власне вставки. Пошук місця для вставки числа в упорядковану послідовність можна виконувати двома способами: послідовним (лінійним) і бінарним (двійковим).

Означення 2. Послідовний (або лінійний) пошук місця елемента в упорядкованій послідовності

Дано впорядковану послідовність a_1, \dots, a_m завдовжки m ($m > 1$) і певне число b . Треба знайти місце числа b у вихідній послідовності, тобто такий номер j ($1 \leq j \leq m + 1$), що число b стане j -м елементом нової впорядкованої послідовності. (У цьому випадку розглядається порядок спадання. Метод легко видозмінюється для зворотного порядку).

Алгоритм послідовного пошуку описано далі.

Упорядкована послідовність переглядається, наприклад, справа наліво. При цьому число, що вставляється, порівнюється послідовно з кожним із чисел послідовності доти, доки знайдеться більше або таке число, що йому дорівнює. Число, що вставляється, має зайняти місце праворуч від нього (рис. 3.4).

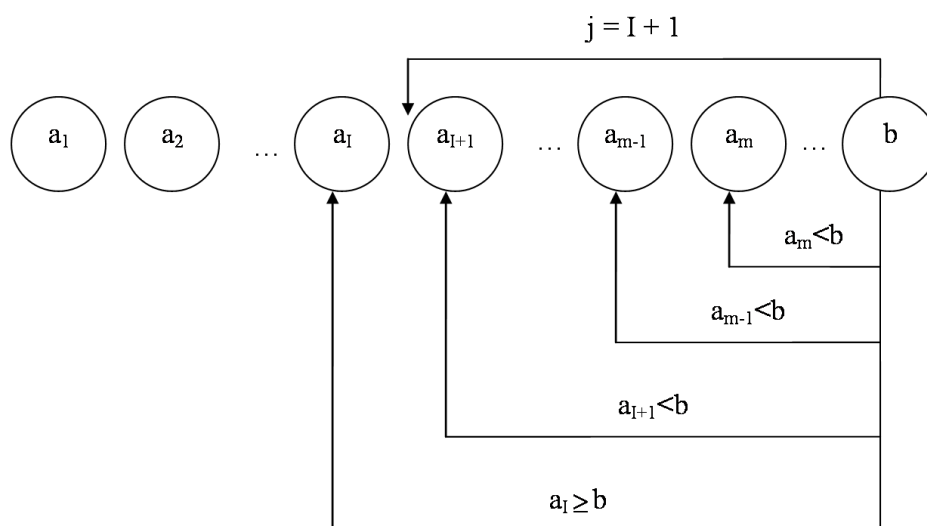


Рисунок 3.4

Якщо $a_m \geq b$, то b має стати $(m + 1)$ -м елементом послідовності, тобто $j = m + 1$. Якщо ж усі числа послідовності менші за число, яке вставляється, то воно має стати першим елементом послідовності, тобто $j = 1$.

Можна розглядати елементи впорядкованої послідовності і зліва направо. Порівняння продовжуються доти, доки знайдеться елемент менший або такий, що дорівнює тому, який вставляється. Число, що вставляється, має зайняти місце ліворуч від нього (рис. 3.5).

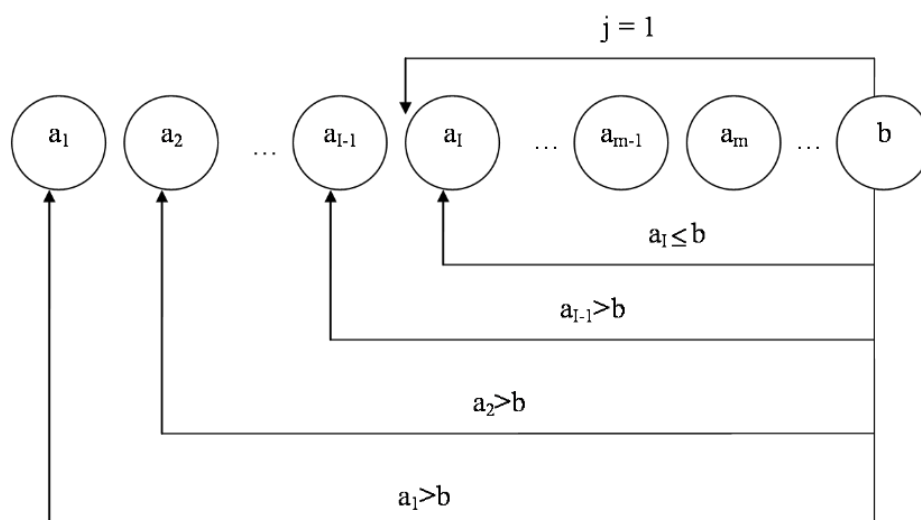


Рисунок 3.5

Якщо елемент, що вставляється, дорівнює або більший за перший (отже, й за усі інші), то він має стати першим елементом, тобто $j = 1$. Якщо ж усі елементи більші за той, що вставляється, то останнім, тобто $j = m + 1$.

Зупинімося для прикладу на перегляді справа наліво. Алгоритм *Пошук(b)* шукає місце числа b в упорядкованій послідовності a_1, \dots, a_m і повертає значення j (рис. 3.6).

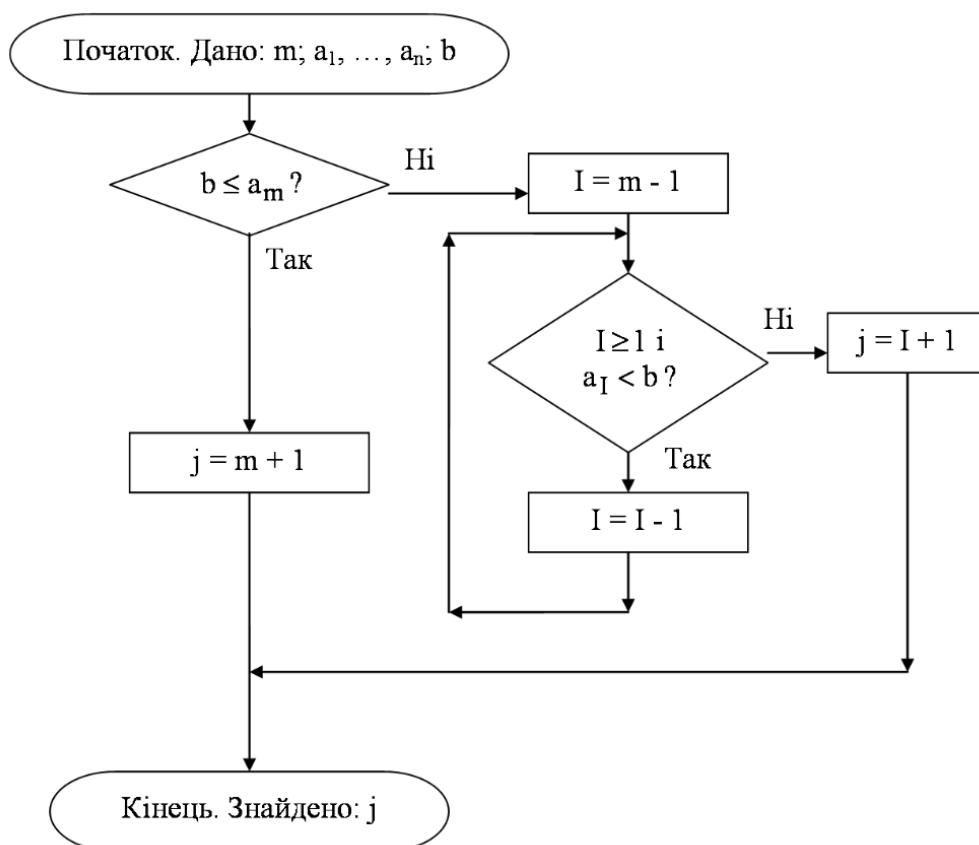


Рисунок 3.6

Зверніть увагу: логічний вираз у циклі використовує операцію I . Якщо хоча б одна з умов стане хибною, цикл завершується. У випадку, коли всі елементи a_1, \dots, a_m менші за b , цикл завершується при значенні $j = 1$. Отже, b має стати першим елементом упорядкованої послідовності.

Існує спеціальний метод, що має змогу уникнути перевірки двох умов на кожному кроці циклу. Він називається **метод бар'єра**. Суть полягає в тому, що вводиться додатковий елемент послідовності a_0 , котрий дорівнює елементу, який вставляється, $-b$. Він служить ніби „бар'єром” перегляду. Хоча б один елемент, більший чи такий, що дорівнює b , тепер знайдеться. Якщо його немає серед a_1, \dots, a_m , то це a_0 . Наявність „бар'єра” забезпечує вихід з циклу (рис. 3.7).

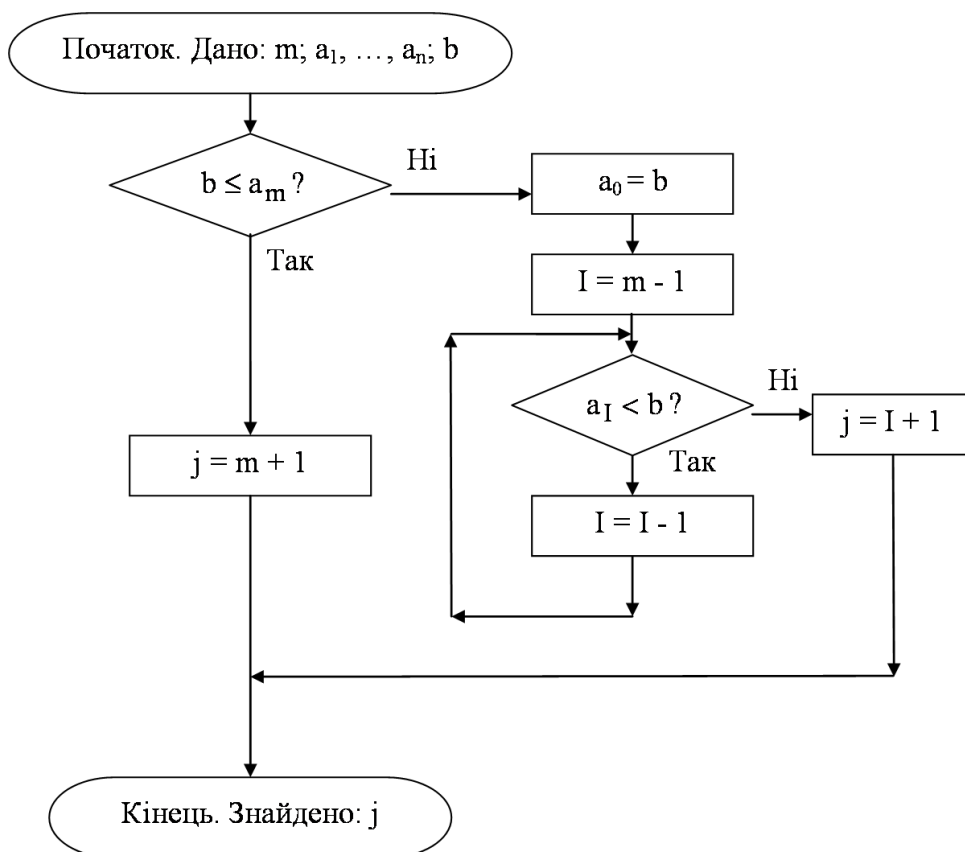


Рисунок 3.7

Що краще: дві умови чи бар'єр? У першому випадку збільшується кількість операцій, у другому – витрачається додаткова пам'ять. Рішення в кожному конкретному випадку залишається за розробником. Воно залежить від можливостей (характеристик) комп'ютера, що використовується для розв'язування задачі.

У методі включень (див. рис. 3.3.) на k -му кроці розв'язується задача пошуку місця елемента a_k в упорядкованій послідовності $a_1, a_k - 1$. Це означає, що розв'язується задача *Пошук (b)* (рис. 3.6) при $m = k - 1$ і $b = a_k$. Оскільки номер k однозначно визначає елемент a_k , відповідний алгоритм назовемо *Пошук (k)* (замість *Пошук (a_k)*). Цей алгоритм відшукує місце елемента a_k в упорядкованій послідовності $a_1, \dots, a_k - 1$ і повертає значення j (рис. 3.8, 3.9).

При використанні бар'єра цей алгоритм має вигляд, наведений на рис. 3.1.10. Метод простих включень, що використовує лінійний пошук, називається методом **простих включень**.

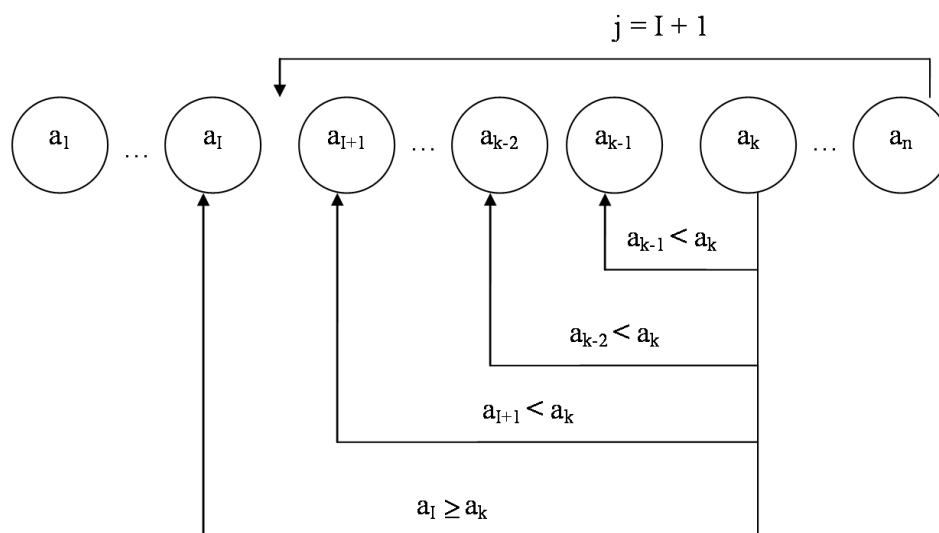


Рисунок 3.8

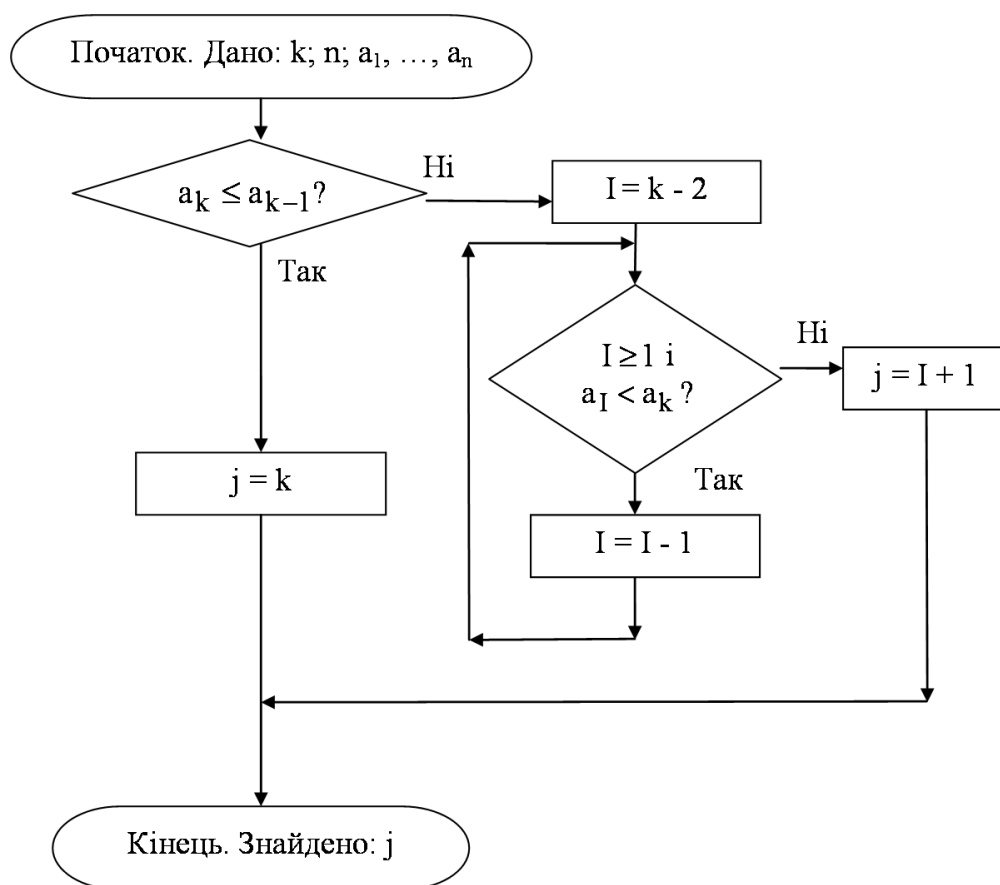


Рисунок 3.9

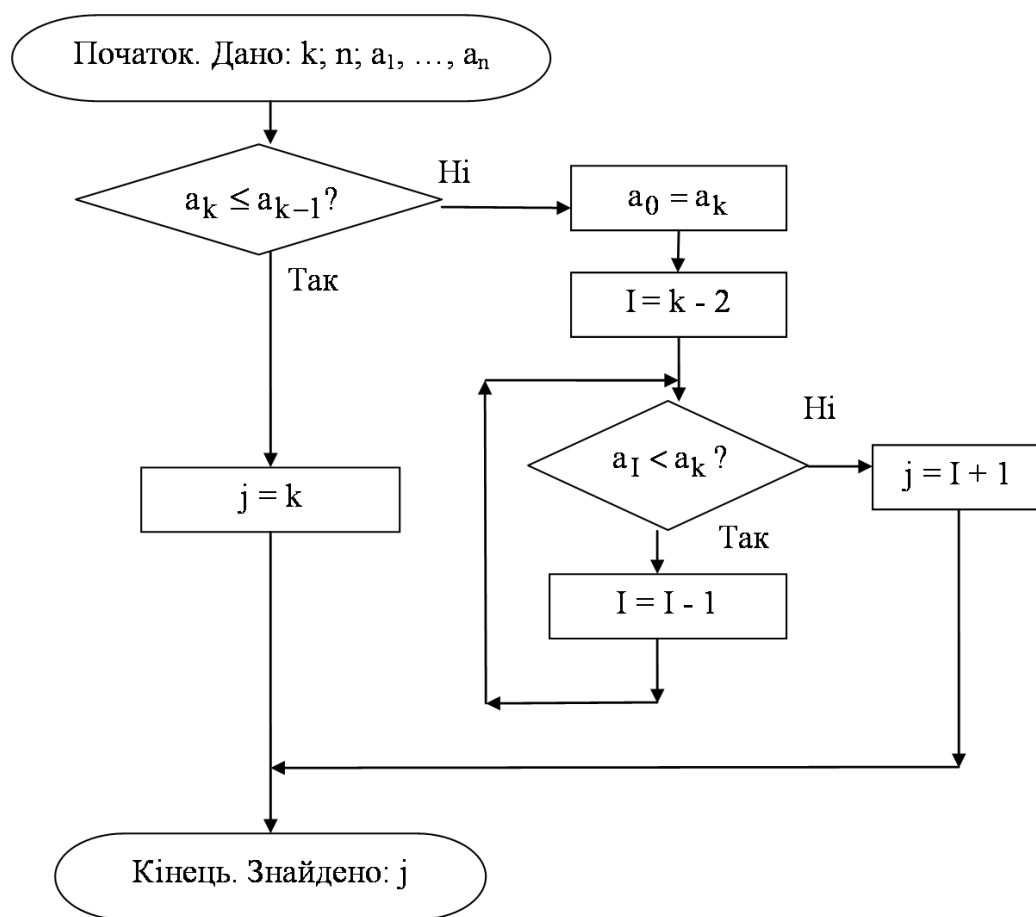


Рисунок 3.10

Означення 3. Бінарний (або двійковий) пошук місця елемента в упорядкованій послідовності Дано впорядковану послідовність чисел a_1, \dots, a_M завдовжки M ($M \geq 1$) і певне число b . Треба знайти місце числа b у вихідній послідовності, тобто такий номер j ($j \leq 1 \leq M + 1$), що число b стане j -м елементом нової впорядкованої послідовності.

Алгоритм бінарного (від англ. *binary* – двійковий) пошуку полягає от у чому. Число, що вставляється, порівнюється із середнім елементом відсортованої (в нашому випадку за спаданням) послідовності (якщо в ній парна кількість членів, то з будь-яким із двох середніх). Якщо число дорівнює середньому елементу, місце знайдено. Якщо ж число менше за середній елемент, то виконується порівняння із середнім елементом правої половини; якщо ж більше, то лівої. Ділення навпіл продовжується доти, доки „є, що ділити”.

Приклад бінарного пошуку місця числа **33** в упорядкованій послідовності чисел **100, 84, 56, 21, 4, 0, -8** наведено на рис. 3.11.

Для запису алгоритму бінарного пошуку введемо дві змінні L , R для збереження відповідно лівої і правої меж розглянутого відрізка послідовності. Спочатку $L = 1$, $R = M$. Пошук ведеться, поки $L \leq R$. Запис $[x]$ позначає цілу частину x . Номер середнього елемента послідовності a_L, \dots, a_R дорівнює $[(L + R)/2]$ (рис. 3.12.).



Рисунок 3.11 - Приклад бінарного пошуку

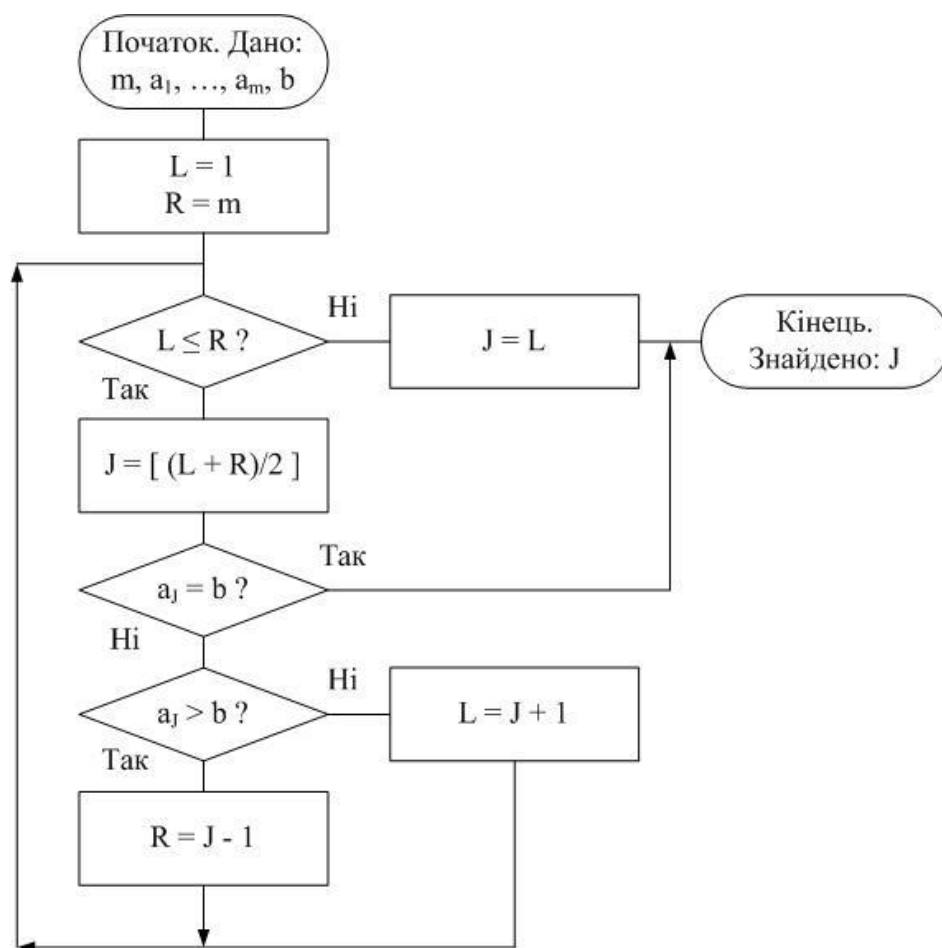


Рисунок 3.12

Зверніть увагу: в наведеному алгоритмі у разі збігу a_j з b вихід із циклу виконується не за умовою закінчення циклу. Багато мов програмування підтримують таку можливість. Якщо ж потрібно забезпечити стандартний вихід із циклу за умовою закінчення, треба штучно його підготувати, наприклад, як показано на рис. 3.13.

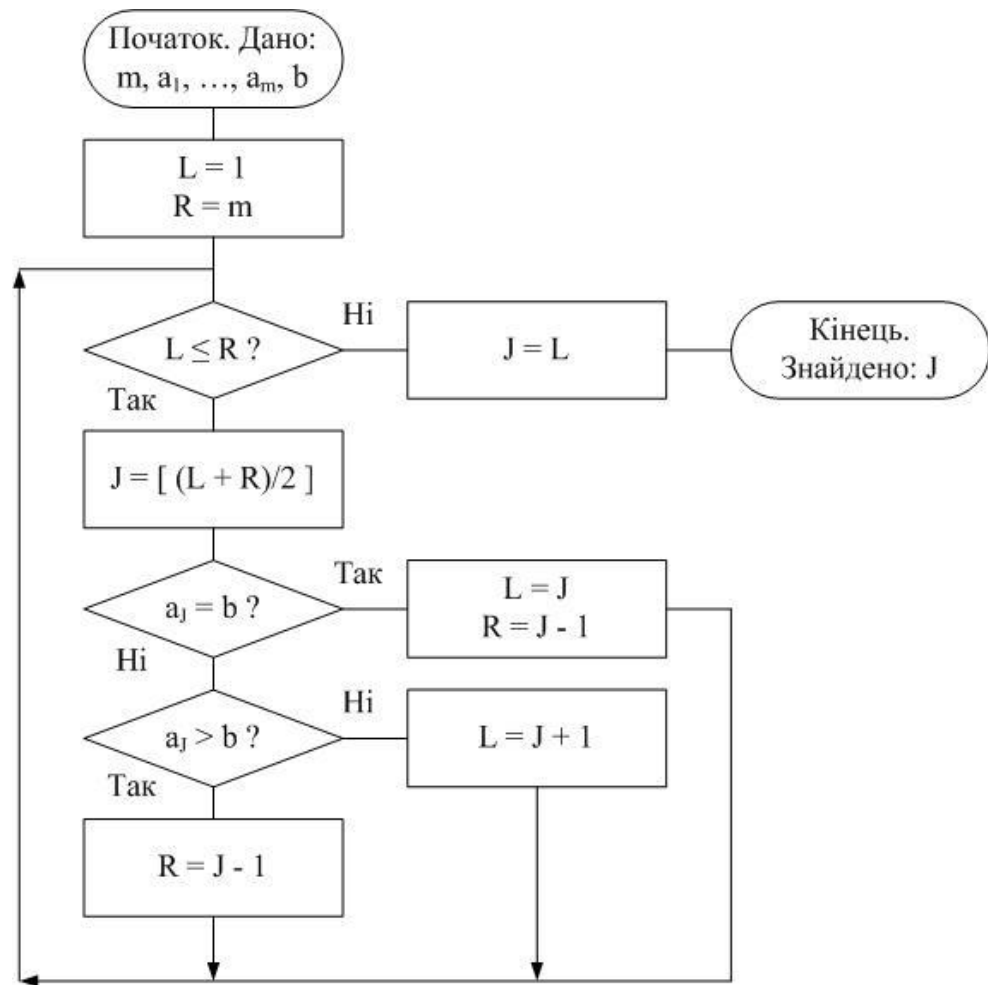


Рисунок 3.13

При використанні бінарного пошуку в методі включень (див. рис. 3.3) алгоритм *Пошук (k)* має вигляд (див. рис. 3.16).

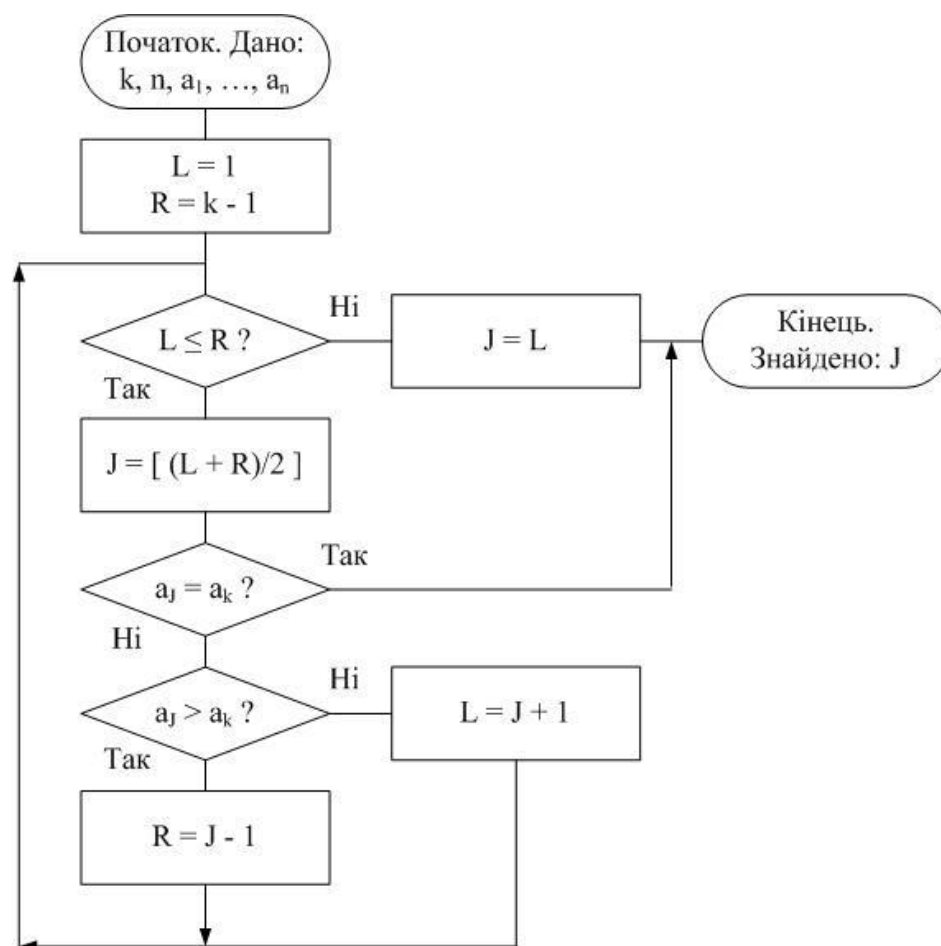


Рисунок 3.14

Метод включень, що використовує бінарний пошук, називається **методом бінарних включень**.

Вставлення елемента у послідовність. На k -му кроці методу включень (див. рис. 3.3) для вставлення елемента a_k на j -те місце ($1 \leq j < k$) використовується алгоритм *Вставка (j)*. Реалізація цього алгоритму залежить від того, яка структура даних використовується для зберігання вихідної послідовності.

Як правило, для елемента, що вставляється, треба «звільнити місце». Для цього всі елементи послідовності справа від a_j , включаючи a_j , слід зсунути на одну позицію вправо, а на місце, що звільнилося, вставити елемент a_k (рис. 3.15).

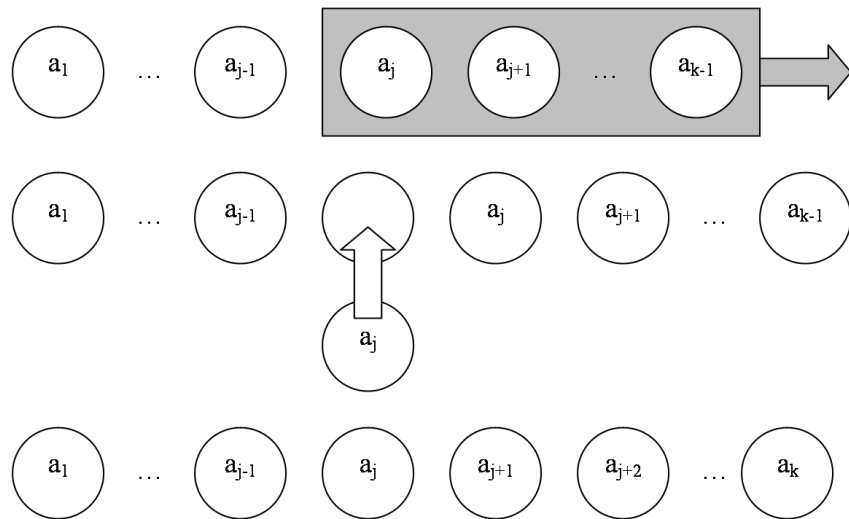


Рисунок 3.15

У такому разі алгоритм Вставка (j) можна записати в такому вигляді, як на рис. 3.16.

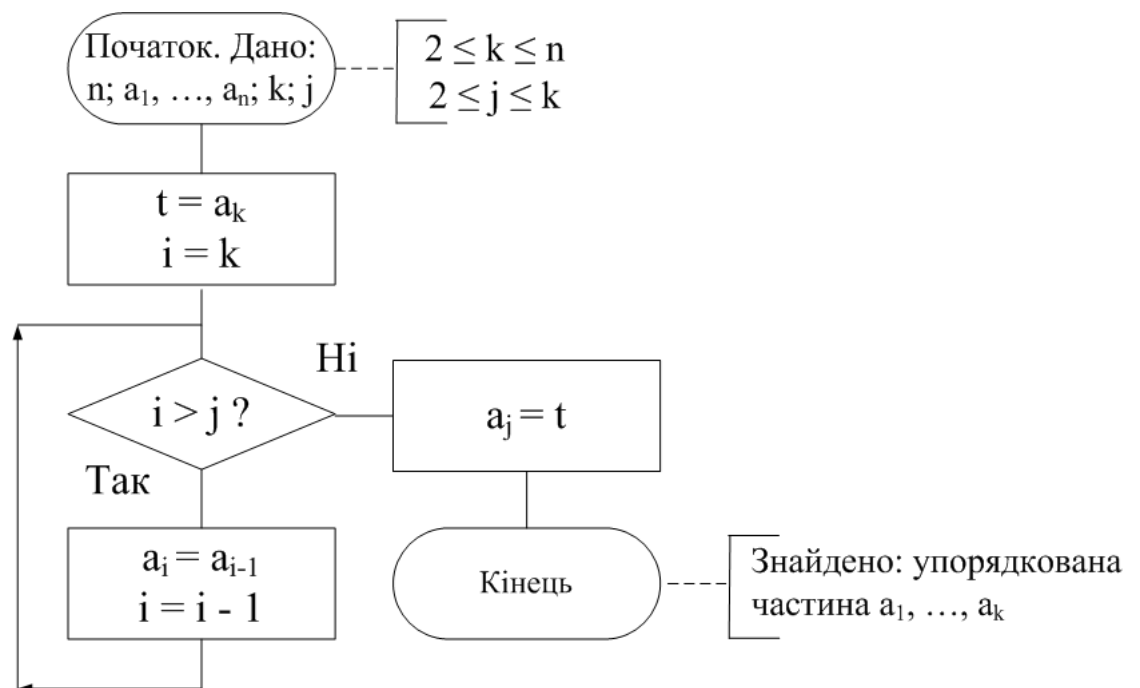


Рисунок 3.16

За посиланням [5] представлено візуалізацію сортування включеннями (АНГЛ. мовою ***Insertion Sort***).

Кнопка *"Randomize array"* генерує числа в послідовності випадковим чином. Кнопка *"Insertion Sort"* запускає сортування. Повзунок *"Animation Speed"* встановлює швидкість роботи анімації.

Кнопка *"Skip Back"* повертає до початкової (невідсортованої) послідовності. Кнопка *"Skip Forward"* показує відсортовану послідовність.

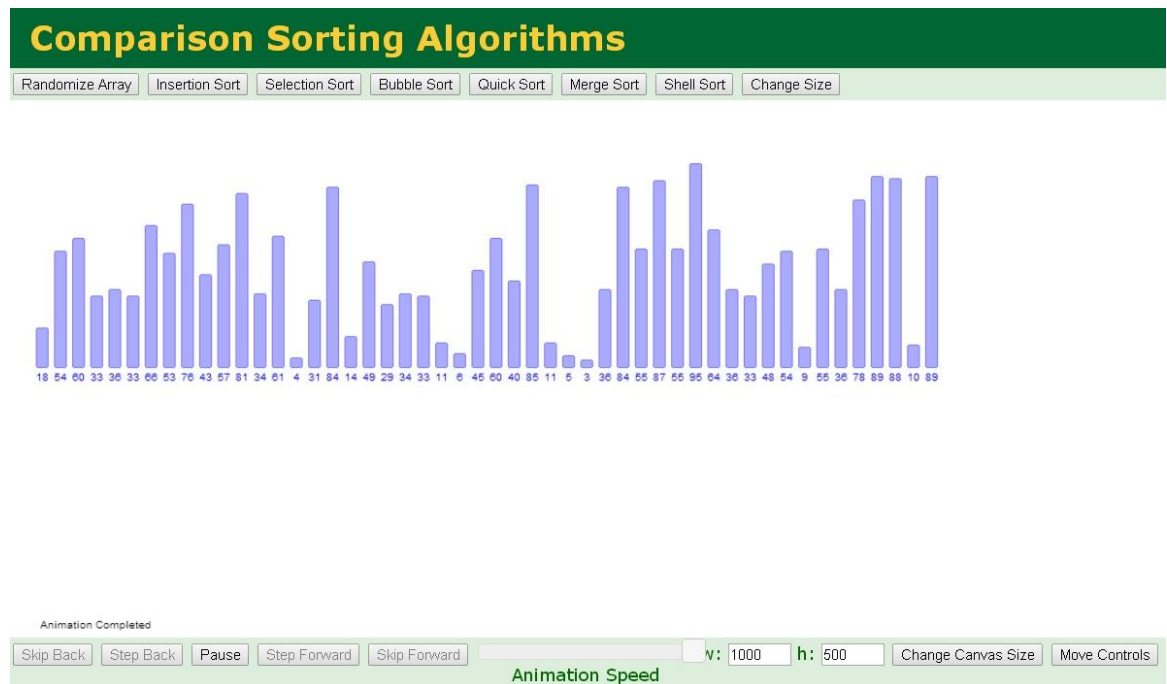


Рисунок 3.17 візуальне представлення сортування

Аналіз трудомісткості методу включень. Проаналізуємо розроблений алгоритм включень. Використання методу послідовного або бінарного пошуку впливає лише на кількість порівнянь. Лінійний пошук:

- мінімальна кількість порівнянь $C_{min} = n - 1$;
- максимальна кількість порівнянь $C_{max} = 0,5n(n - 1)$;
- в середньому кількість порівнянь $C_{сер} = 0,25n^2 + 0,25n + 0,5$.

У випадку бінарного пошуку кількість порівнянь не залежить від початкового порядку елементів. Отже, всього кількість порівнянь $C = \log_2 1 + \log_2 2 + \dots + \log_2 (n - 1)$.

Ця сума менша, ніж $n \log_2 n$.

Для більш строгих оцінок потрібні спеціальні знання. Щодо кількості пересилань M , вона залежить від конкретної реалізації алгоритму вставлення. У будь-якому випадку M зростає не швидше, ніж n^2 .

3.2. Алгоритмізація задачі за темою роботи

На стартовому вікні виводиться назва тренажеру, інформація щодо розробника та теми. Розташовано дві кнопки перша це перехід до тестування а друга на дистанційний курс, для того щоб перейти до читання дистанційного курсу потрібно бути зареєстрованим на сайті дистанційної освіти.

Після натиснення на кнопку «Тренажер» відбувається перехід до нового вікна програми з питаннями та закриття головної сторінки.

Основні означення з теми

Крок 1. Виберіть відповідь. В якому напрямі може виконуватися сортування:

- з ліво на право;
- з права на ліво;
- в обох напрямках;
- Немає правильної відповіді.

Якщо вказана неправильна відповідь то користувачу відображається помилка та продовжується робота з програмою.

Крок 2. Виберіть відповідь. Заповніть пусті поля «Алгоритм Пошук(b)»

- $j = a+b$;
- $j = b+m$;
- $j = I+m$;
- $j = m+1$.

Якщо вказано неправильну відповідь, то відображається помилка, після виправлення на правильну відповідь відбудеться перехід до наступного кроку.

Крок 3. Виберіть правильну відповідь в ComboBox «Метод бар'єра»:

- $a_0 = b$;

- $a_0 > b$;
- $a_0 \leq b$;
- $a_0 \geq b$.

Якщо відповідь не правильна то буде відображена помилка, та продовжена робота з тренажером, після вказання правильної відповіді відбудеться перехід до наступного питання.

Крок 4. Виберіть правильну відповідь в ComboBox «Метод бар'єра»

- $a_1 \geq b?$;
- $a_1 > b?$;
- $a_1 < b?$.

Якщо відповідь не правильна то буде відображена помилка, та продовжена робота з тренажером, після вказання правильної відповіді відбудеться перехід до наступного питання.

Крок 5. Введіть правильні відповіді в «Бінарний пошук місця елемента»:

- $L = ?$;
- $R = M = ?$;
- $(L + R)/2 = ?$.

Після вказання правильної відповіді відбудеться перехід до наступного питання. У разі неправильної відповіді з'явиться вікно помилки та підказки.

Крок 6. Виберіть правильну відповідь в елементах ComboBox

В даному кроці необхідно послідовно обрати правильні відповіді в елементах «ComboBox»

Після вказання правильної відповіді відбудеться перехід до наступного питання. У разі неправильної відповіді з'явиться вікно помилки та підказки.

Крок 7. Метод "Двох умов" відрізняється від методу "Бар'єра" тим що -?:

- збільшується кількість операцій;
- витрачається додаткова пам'ять.

Після вказання правильної відповіді відбудеться перехід до наступного питання. У разі неправильної відповіді з'явиться вікно помилки та підказки.

Крок 8. Аналіз трудомісткості методу включень:

- $C_{min} = ?$;
- $C_{max} = ?$;
- $C_{сер} = ?$.

Після поетапного вказання правильних відповідей з'являється вікно що ви пройшли всі питання та можливість перейти до головного вікна програми. У разі неправильної відповіді з'явиться вікно помилки та підказки.

3.3. Блок-схема алгоритму

На рис. 3.3.1 зображена блок-схема алгоритму для першого питання.

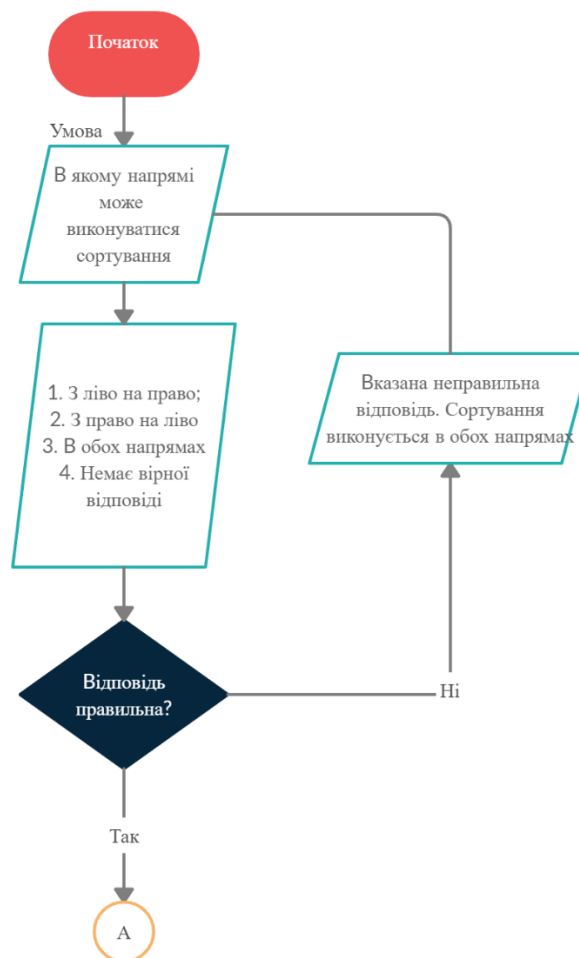


Рисунок 3.18 – Блок схема

3.4. Обґрунтування вибору програмних засобів для реалізації завдання роботи

Для розробки тренажеру обрана мова програмування C#[9]. C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Синтаксис C# подібний до Java та C++. Мова має строгу статичну типізацію, підтримує поліморфізм, вказівники на функції-члени класів, атрибути, перевантаження операторів, властивості, події, винятки, коментарі у форматі XML. Переїнявши багато від своїх попередників — мов C++, Object Pascal, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, мова C#, на відміну від C++, не передбачає множинне успадкування класів.

Для створення графічного інтерфейсу обрано технологію Windows Form. Windows Forms — інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді.

Було обрано середовище розробки програмного забезпечення Microsoft Visual Studio — серія програмних продуктів від компанії Microsoft, які включають інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms та багато інших. Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторинга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення

створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних.

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення можливостей практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального моделювання коду на мові) предметно-орієнтованих мов програмування) або інструментів для іншого процесу розробки програмного забезпечення (наприклад, Team Explorer для роботи з Team Foundation Server).

В зв'язку з тим що обрані засоби відповідають всіми потрібними мені методами та відповідають моїм навичкам було обрано саме це середовище розробки програмного забезпечення та мова програмування.

РОЗДІЛ 4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис процесу програмної реалізації

Розглянемо, як створювався тренажер. Першим кроком створення самого проекту за для цього на головному вікні програми Visual Studio обираємо «Создание проекта» (рис 4.1).

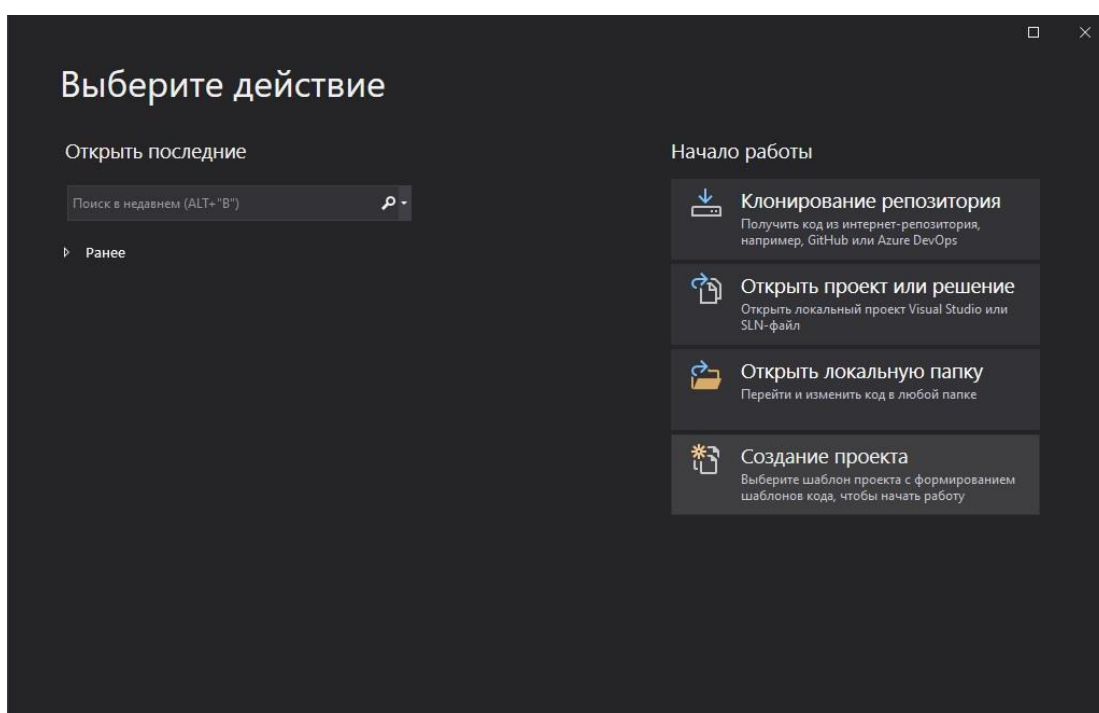


Рисунок 4.1 – Головне вікно.

Після натиснення на кнопку відкриється вікно в якому ми обираємо мову програмування C# та «Приложение Windows Forms» (рис 4.2). Після натиснення на кнопку «Далее» необхідно вказати назву проекту його розміщення та обрати платформу (рис 4.3).

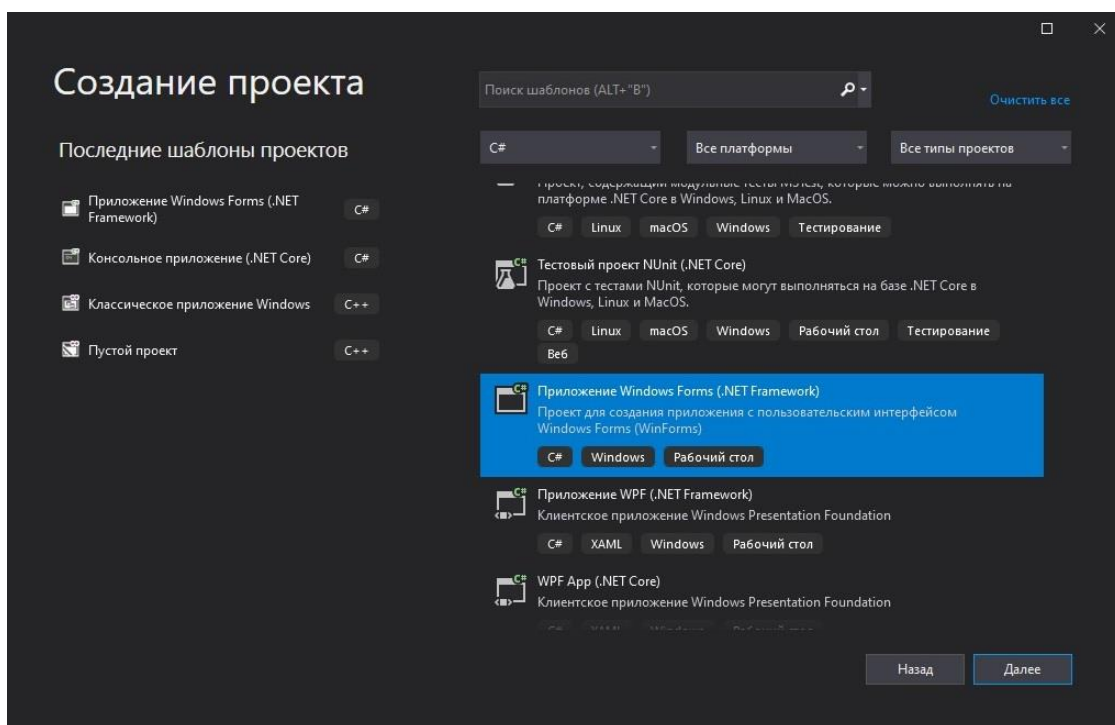


Рисунок 4.2 – Вікно створення проекту.

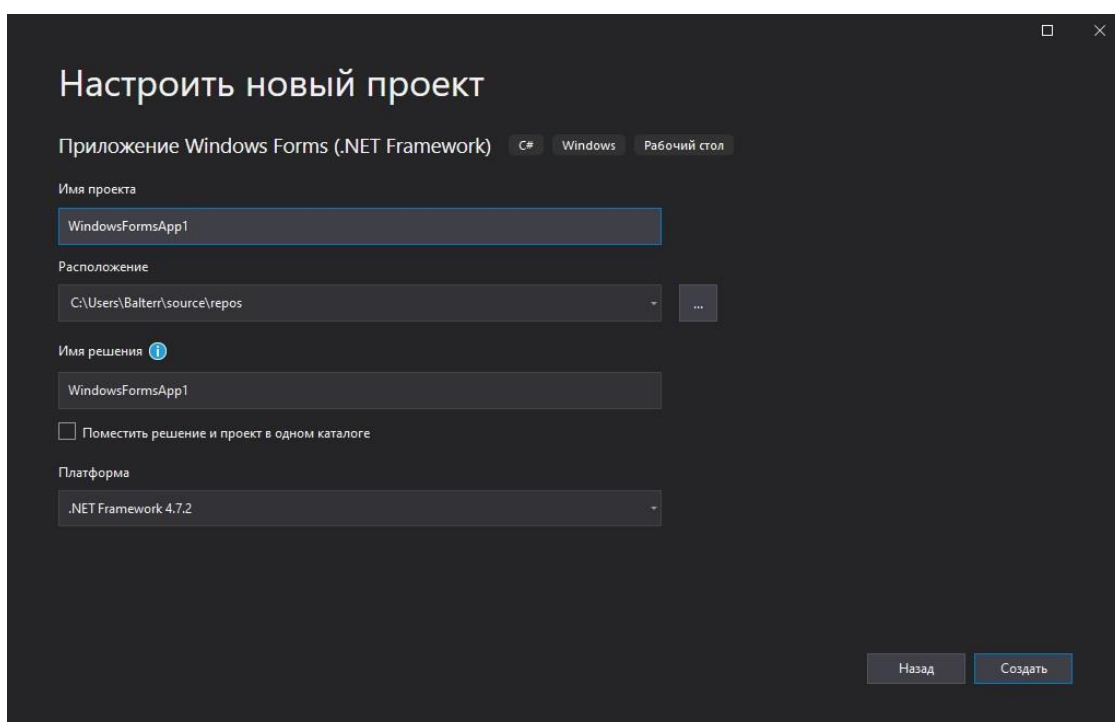


Рисунок 4.3 – Вікно налаштувань.

Після створення проекту ми отримуємо пусту форму (рис 4.4) на яку ми додаємо елементи використовуючи «Панель елементів» за допомогою якої ми додаємо кнопки, текст, текстові поля, картинки.

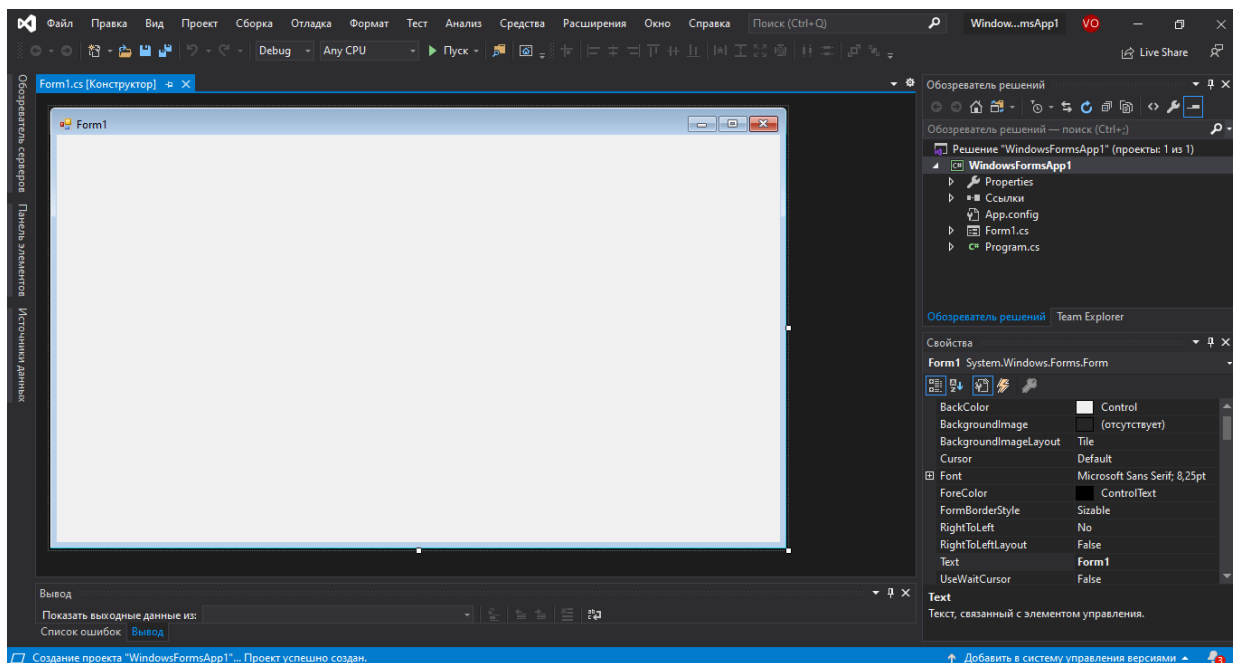


Рисунок 4.4 – Вікно з пустою формою.

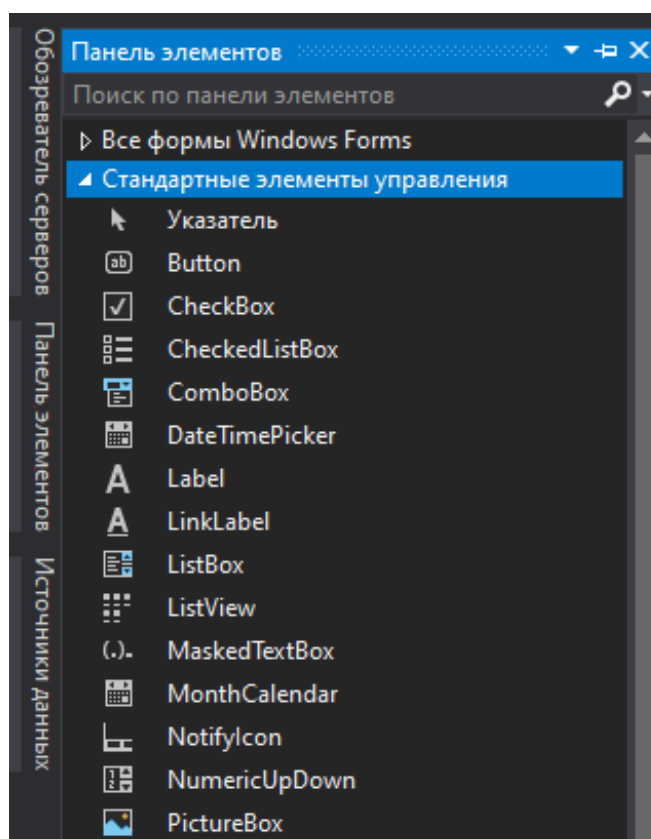


Рисунок 4.5 – Панель елементів.

Колір форми, назву та інші налаштування можна виставити за допомогою панелі «Свойства» яка по замовчуванню розміщена в правій нижній частині програми.

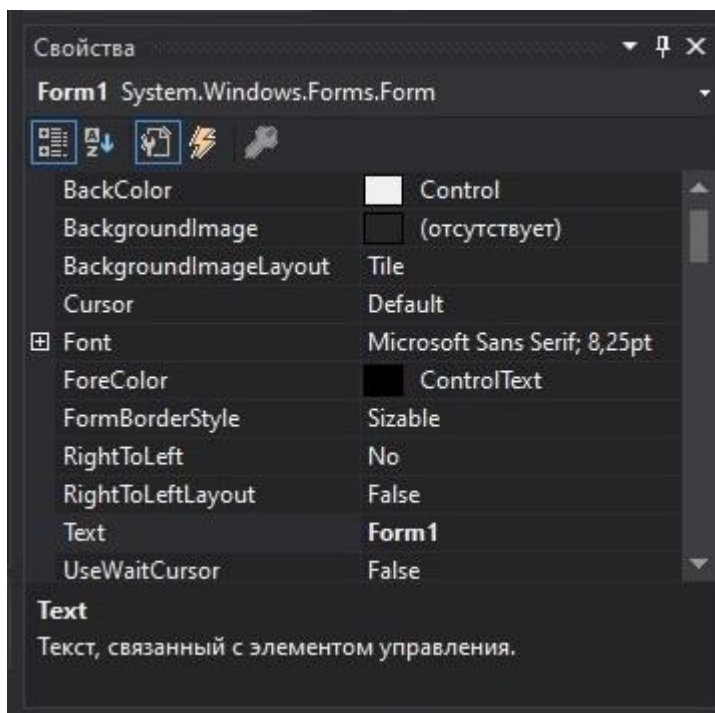


Рисунок 4.6 – Панель налаштувань.

Після того як ми розмістили деякі елементи на формі, ми маємо написати код для них, для цього достатньо натиснути два рази на цей елемент та відкриється вікно редагування в якому автоматично створеться основа коду для нього.

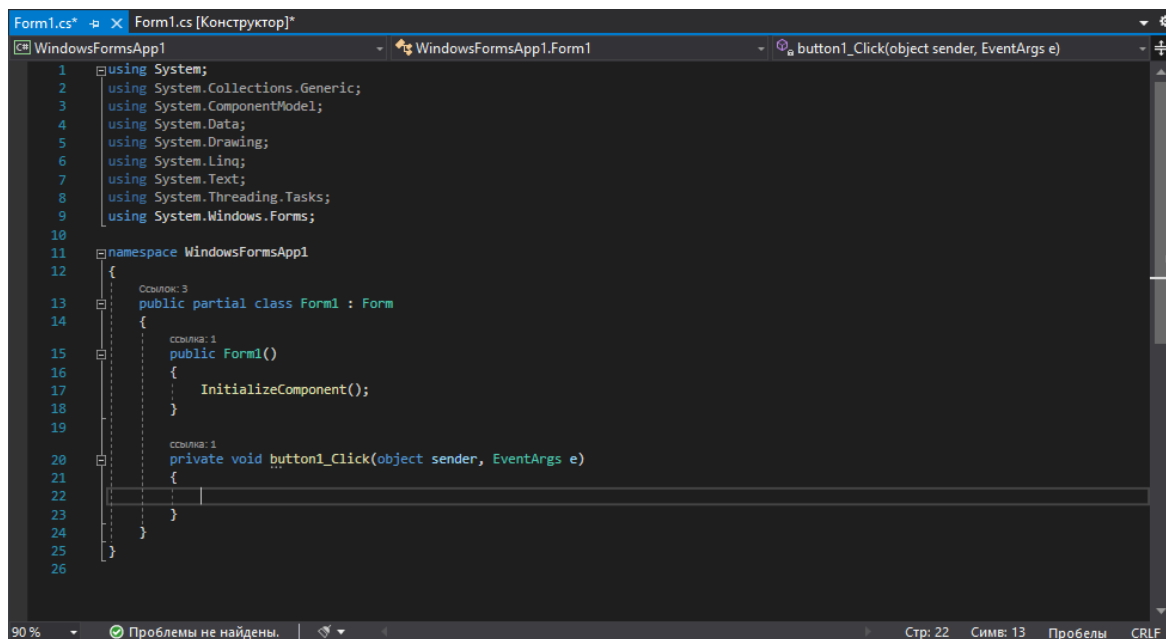


Рисунок 4.7 – Панель елементів.

Для розміщення малюнків було використано компонент PictureBox з «Стандартних елементів управління» панелі елементів Windows Form (рис. 4.5).

Заздалегідь зображення були завантажені з дистанційного навчального курсу «» та збережені на локальному диску. Зображення були завантажені з локального ресурсу та розміщені на формі(рис. 4.7).

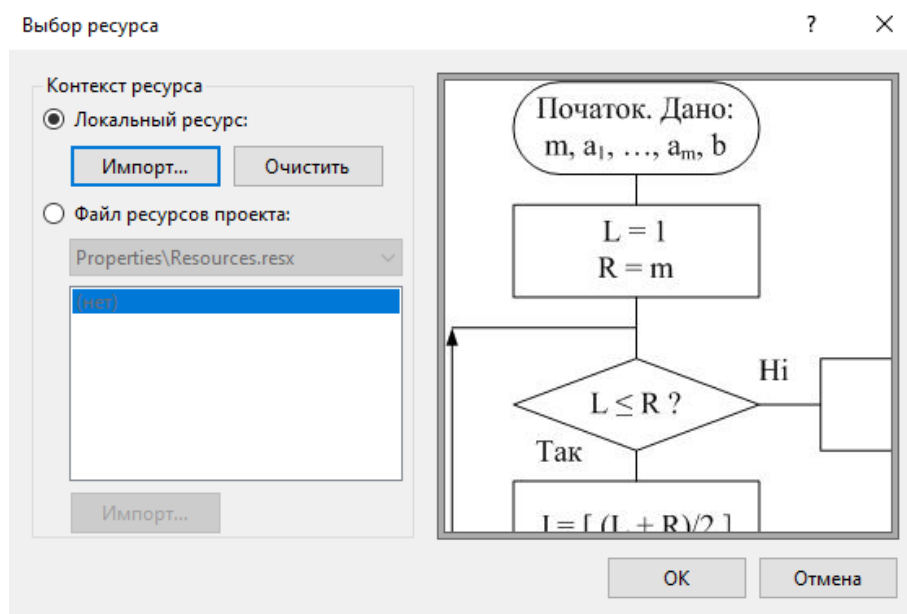


Рисунок 4.8 – Завантаження малюнку.

Далі було виставлено розмір картинок (властивість Size) та розміщена локація картинок (властивість Location).

Дальше ми розглянемо код деяких з елементів програми . Для кнопки «Наступне питання» було створено код:

```
public partial class Form6 : Form
{
    public Form6()
    {
        InitializeComponent();
        string[] sort = { "J = L", "L = J + 1", " J = L + 1", "L = J - 1" };
        string[] sort1 = { "aj = b?", "aj < b?", "aj > b?", "aj <= b?" };
        comboBox1.Items.AddRange(sort);
        comboBox3.Items.AddRange(sort);
        comboBox2.Items.AddRange(sort1);
        comboBox4.Items.AddRange(sort1);
        comboBox2.Enabled = false;
        comboBox3.Enabled = false;
        comboBox4.Enabled = false;
        button2.Visible = false;
        button3.Visible = false;
        button4.Visible = false;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        if (comboBox3.SelectedIndex == 1)
```

```

    {
        comboBox2.Enabled = true;
        button2.Visible = false;
        button3.Visible = true;
    }
    else
    {
        MessageBox.Show("Вказана неправильна
відповідь" + "\r\n" + "J = L");
    }
}

private void button1_Click(object sender, EventArgs e)
{
    if (comboBox1.SelectedIndex == 2)
    {
        comboBox3.Enabled = true;
        button1.Visible = false;
        button2.Visible = true;
    }
    else
    {
        MessageBox.Show("Вказана неправильна
відповідь" + "\r\n" + "L = J + 1");
    }
}

```

Проаналізуємо його.

Данні які виводяться в елементі ComboBox зберігаються в масиві та мають свій певний індекс. Поки не буде вказана вірна відповідь в першому елементі ComboBox всі наступні не будуть доступні для надання відповіді.

Якщо в ComboBox обрано не вірну відповідь, то з'являється повідомлення з помилкою та вірною відповіддю .

Якщо відповідь в першому елементі ComboBox вірна, то дозволяється редагування наступного елементу ComboBox, після того як в всіх елементах дана правильна відповідь відбувається перехід на наступний крок шляхом відкриття іншої форми.

Розглянемо інший крок.

```
public Form5()
{
    InitializeComponent();
    textBox2.Enabled = false;
    textBox3.Enabled = false;
    button2.Visible = false;
    button3.Visible = false;
    button4.Visible = false;
}
private void button1_Click_1(object sender, EventArgs
e)
{
    int q1 = int.Parse(textBox1.Text);
    if (q1 == 1)
    {
        button2.Visible = true;
        textBox2.Enabled = true;
        button1.Visible = false;
```

```

    }
    else
    {
        MessageBox.Show("Вказана                неправильна
відповідь" + "\r\n" + "L = 1");
    }
}

private void button2_Click(object sender, EventArgs e)
{
    int q2 = int.Parse(textBox2.Text);
    if(q2 == 7 ) {
        textBox3.Enabled = true;
        button2.Visible = false;
        button3.Visible = true;
    }
    else
    {
        MessageBox.Show("Вказана                неправильна
відповідь" + "\r\n" + "R = M = кількість елементів масиву =
7");
    }
}

private void button3_Click(object sender, EventArgs e)
{
    int q3 = int.Parse(textBox3.Text);
    if (q3 == 4)
    {

```

```

        button3.Visible = false;
        Form6 f = new Form6();
        Hide();
        f.ShowDialog();
    }
    else
    {
        MessageBox.Show("Вказана                неправильна
відповідь" + "\r\n" + "(R + L)/2 = (1 + 7)/2 = 4");
    }
}

```

Проаналізуємо код.

При натисненні кнопки «Наступне питання» відбувається перевірка правильна відповідь чи ні.

Якщо відповідь вірні, дозволяється редагування іншого елементу TextBox.

Якщо у відповідях є помилки, то з'являється відповідь на питання.

В разі якщо в всіх елементах TextBox вказали правильні відповіді відбувається перехід на наступну форму з питаннями.

4.2. Інструкція по роботі з програмою

Після запуску програми відкриється головне вікно, на ньому вказана основна інформація, та розташовано дві кнопки. При натисненні на кнопку «Дис. Курс» відкриється заданий по замовчуванні браузер з дистанційним курсом по «Сортування включеннями» в іншому випадку ви переходите до першого кроку тренажера. В разі неправильної відповіді з'являється вікно з помилкою та правильна відповідь.

При натисканні на кнопку «Тренажер» (рис 4.9) відбудеться перехід до вікна з питаннями(рис 4.10).

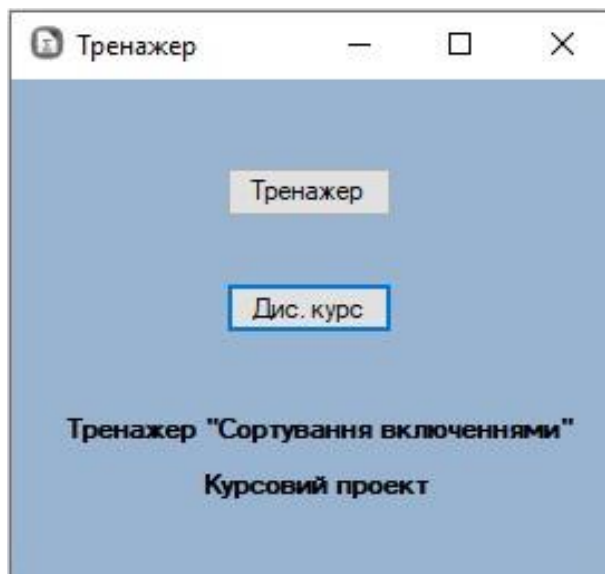


Рисунок 4.9 – Головне вікно програми.

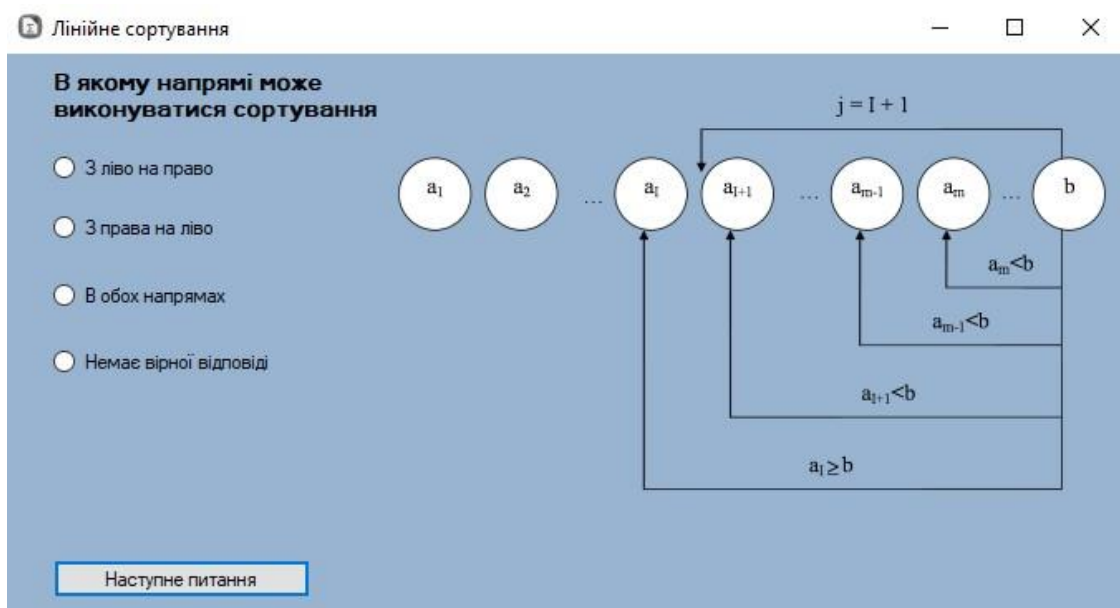


Рисунок 4.10 – Вікно з першим питанням.

Якщо вказана не правильна відповідь то з'явиться вікно з помилкою.

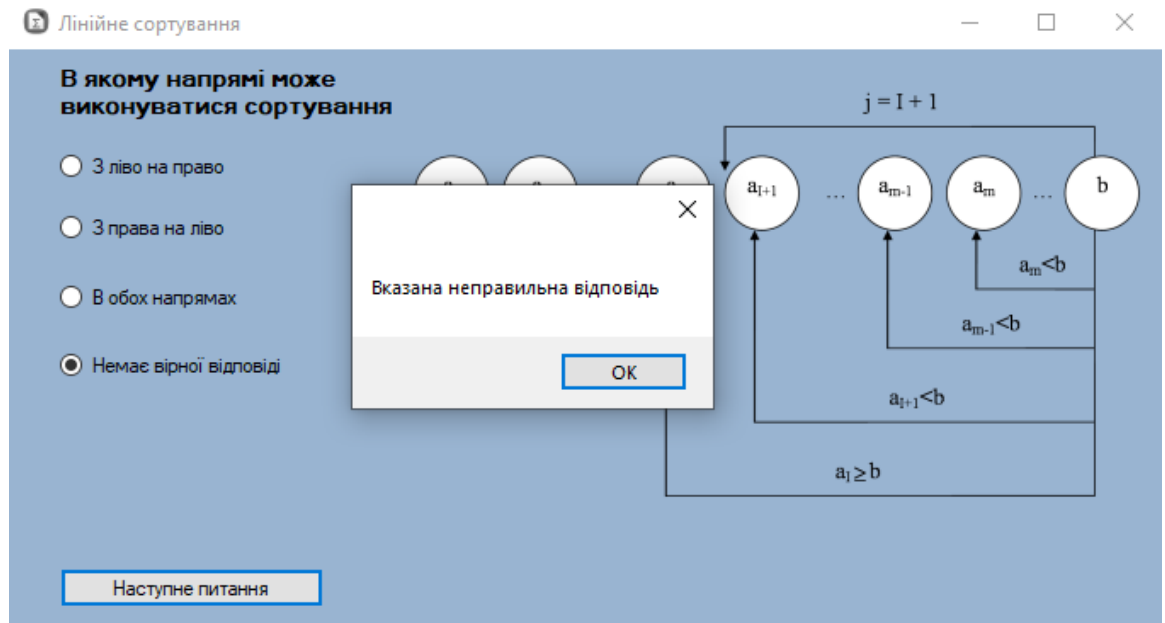


Рисунок 4.11 – Вікно помилки.

В іншому випадку відбудеться перехід до другого питання відповіді на які потрібно обрати в полі «ComboBox»

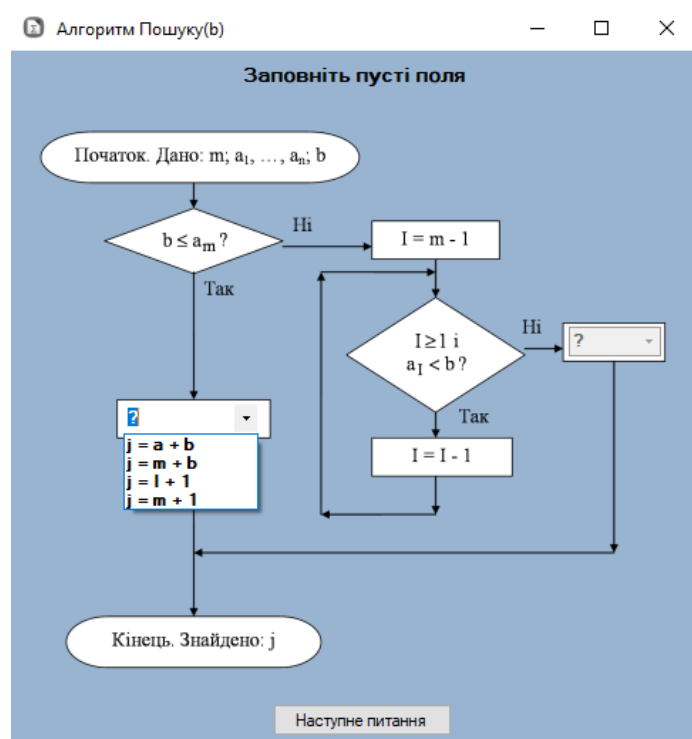


Рисунок 4.12 – Вікно з відповідями на питання в «ComboBox».

Якщо відповіді вірні відбувається перехід до наступного третього питання відповіді на які необхідно дати в елементі програми «ComboBox» поетапно. В іншому випадку з'явиться вікно з помилкою та підказкою на даний етап.

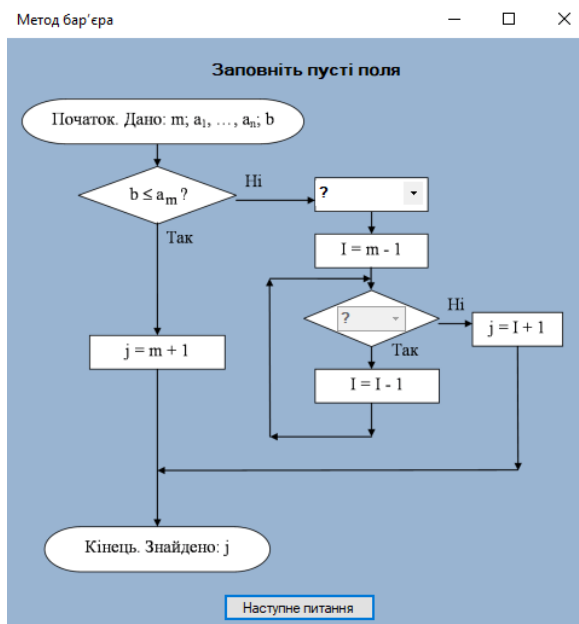


Рисунок 4.13 – Третє питанняю.

Після цього відкриється вікно з четверта форма з питаннями відповіді на які необхідно поетапно давати в полях «TextBox». Вразі неправильної відповіді з'явиться вікно з помилкою.(рис 4.15)

Бінарний пошук місця елемента

Дано масив: A = (1, 11, 23, 9, 4, 7, 5)

Знайти

L = R = M =

Номер середнього елемента послідовності
(L + R) / 2 =

Рисунок 4.14 – Вікно з відповідями на питання в «TextBox».

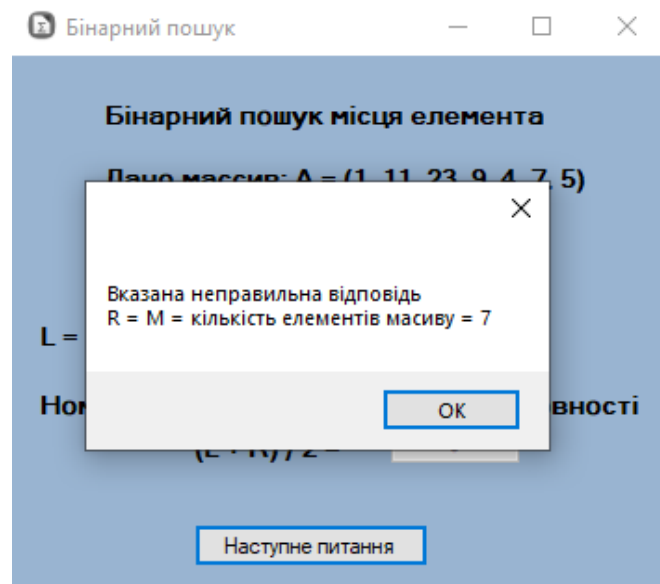


Рисунок 4.15 – Вікно помилки.

Далі відбувається перехід до наступного етапу тренінгу а саме до п'ятої форми з питаннями. На ній розташовано чотири елемента «ComboBox» в яких поетапно потрібно обрати відповіді. В разі невірної відповіді з'являється помилка та підказка на кожну неправильну відповідь.

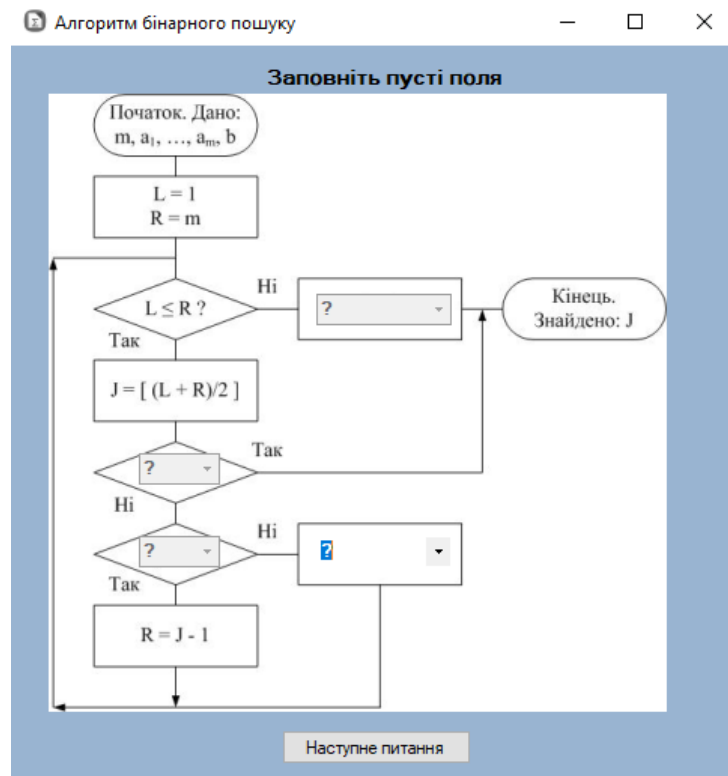


Рисунок 4.16 – Вікно з п'ятим етапом

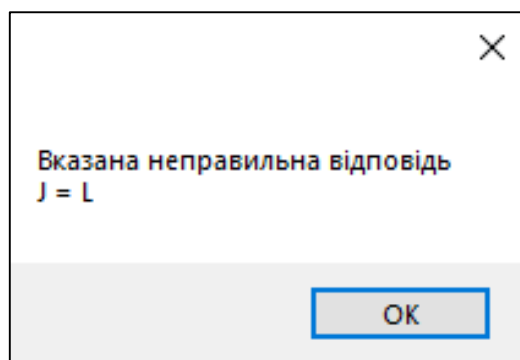


Рисунок 4.17 – Вікно помилки на одне питання п'ятого етапу.

В разі правильної відповіді відбувається перехід до наступного етапу в якому необхідно обрати одну із декількох відповідей

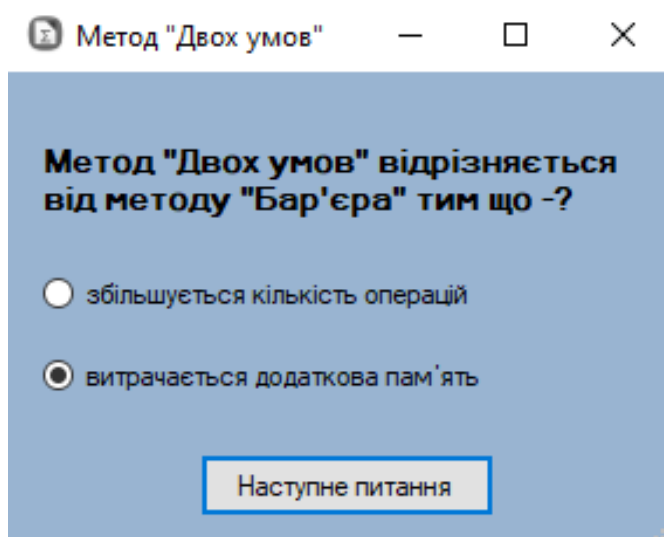


Рисунок 4.18 – Вікно з шостим етапом

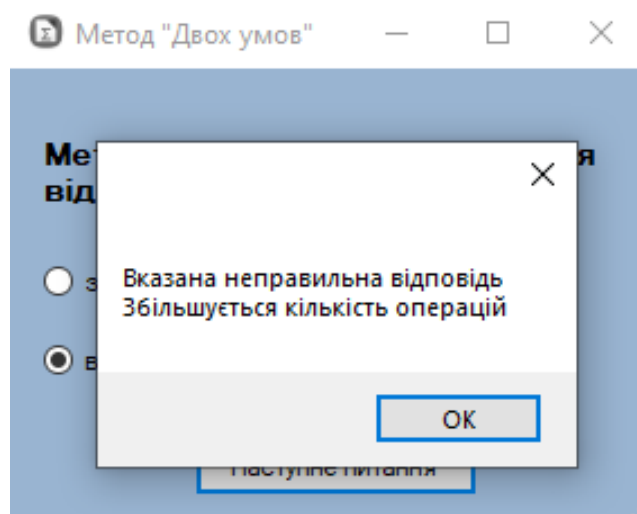


Рисунок 4.19 – Вікно помилки

На наступному етапі необхідно знайти трудомісткості методу включень та відповісти на три питання практично. В разі неправильної відповіді відкриється вікно з помилкою та підказкою. Після надання правильних відповідей на питання з'явиться вікно що ви пройшли всі питання та кнопка для виходу на головне вікно.

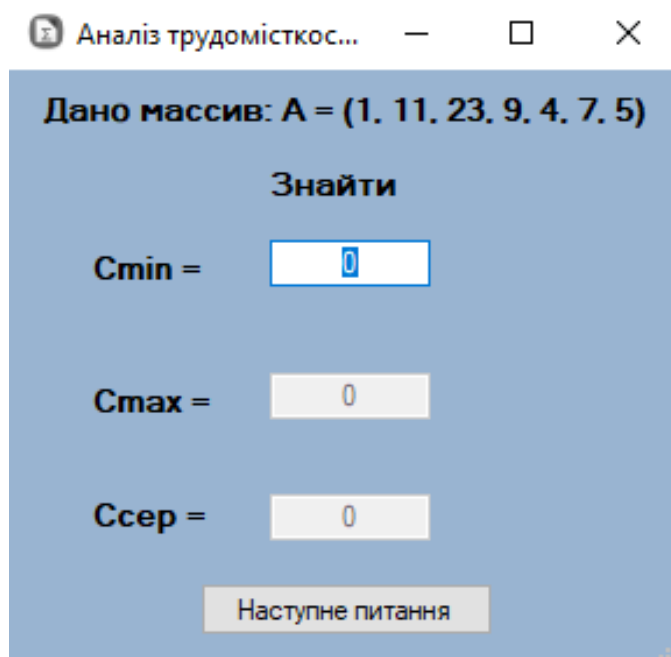


Рисунок 4.20 – Вікно питання

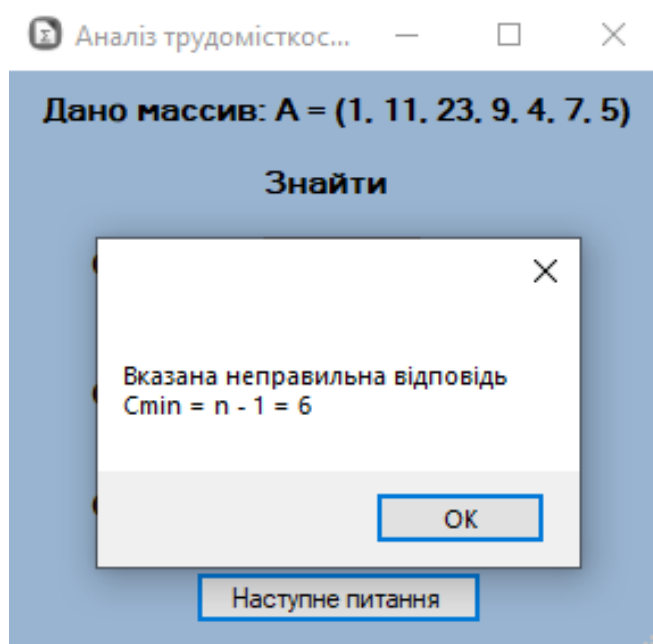


Рисунок 4.21 – Вікно помилки

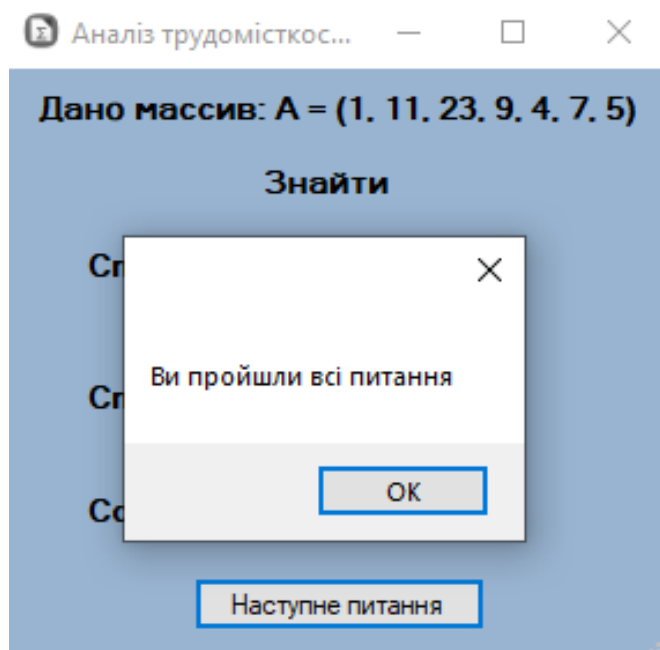


Рисунок 4.22 – Повідомлення про закінчення питань

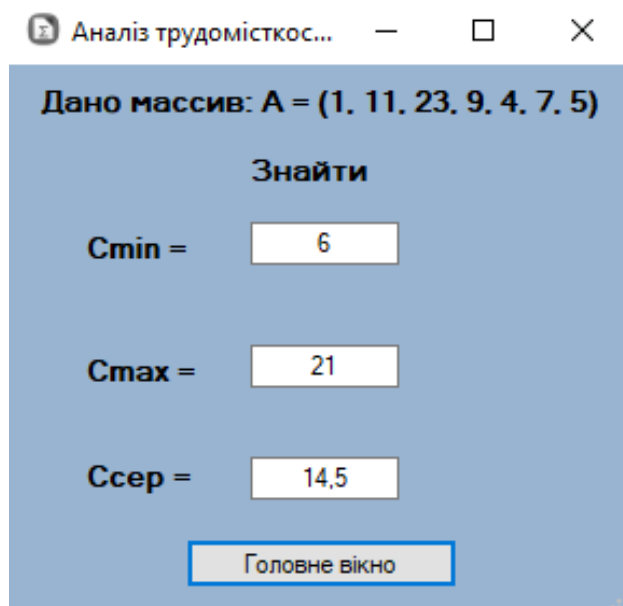


Рисунок 4.23 – Кнопка переходу до головного вікна

4.3. Перевірка валідності

Тестування проводилось на персональних комп'ютерах з операційною системою Windows 7, 10.

Тест-кейс № 1. Запуск програми.

- запустити програму;
- перевірити працездатність всіх доступних елементів;
- натиснути кнопку «Вихід».

Очікуваний результат

Програма запуститься на всіх представлених ОС та всі елементи працездатні.

Фактичний результат

Програма запустилась на всіх представлених ОС та всі елементи працездатні.

Тест-кейс № 2. Перехід до питань.

- запустити тест;
- перевірити працездатність представлених елементів;
- обрати правильну відповідь;
- натиснути кнопку «Наступне питання».

Очікуваний результат

Програма запуститься на всіх представлених ОС, відбувається перехід до наступного питання , всі елементи працездатні.

Фактичний результат

Програма запуститься на всіх представлених ОС, відбувається перехід до наступного питання , всі елементи працездатні.

Тест-кейс № 3. Перевірка вікна з помилкою.

- запустить програму;
- перейти будь якого питання;
- допустити помилку.

Очікуваний результат

Після неправильної відповіді з'явиться вікно з помилкою та підказкою.

Фактичний результат

Після неправильної відповіді з'явиться вікно з помилкою та підказкою.

Тест-кейс № 4. Перевірка працездатності програми після неправильної відповіді.

- запустить програму;
- перейти питання;
- допустити помилку;
- виправити помилку на правильну відповідь;
- натиснути на кнопку «Наступне питання».

Очікуваний результат

Після виправлення помилки на правильну відповідь та натиснення на кнопку «Наступне питання» відбудеться перехід до іншого тесту або етапу тесту.

Фактичний результат

Після виправлення помилки на правильну відповідь та натиснення на кнопку «Наступне питання» відбудеться перехід до іншого тесту або етапу тесту.

Тест-кейс № 5. Перевірка працездатності кнопки «Головне вікно».

- запустить програму;
- перейти питання до останнього питання;
- дати правильну відповідь на питання;
- натиснути на кнопку «Головне вікно»;
- закрити програму.

Очікуваний результат

Після правильної відповіді з'явиться кнопка «Головне вікно» при натисненні на неї відбудеться перехід до головного вікна програми.

Фактичний результат

Після правильної відповіді з'явилась кнопка «Головне вікно» при натисненні на яку відбувся перехід до головного вікна програми..

Під час тестування фатальних помилок не було виявлено.

ВИСНОВКИ

Робота посвячена розробці тренажеру з теми «Сортування включеннями з лінійним та бінарним пошуком» дистанційного навчального курсу «Алгоритми та структури даних» та розробка його програмного забезпечення. Результатом проведеної курсової роботи дають підставу зробити такі висновки:

Робота була розділена на дві частини.

В першій частині розглянуто весь необхідний теоретичний матеріал, зокрема лінійний та бінарний пошук.

В другій частині розроблено алгоритм роботи програми та безпосередньо сам тренажер.

Для створення тренажеру обрано мову програмування C# та інтегроване середовище розробки Visual Studio.

Тренажер протестовано методом тест кейсу, усі наявні помилки усунені.

СПИСОК ЛІТЕРАТУРИ

1. Ємець О. О. Методичні рекомендації щодо оформлення пояснювальних записок до курсових проектів (робіт) для студентів за освітньою програмою «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки та інформаційні технології», «Комп'ютерні науки» галузь знань - 12 «Інформаційні технології» / О. О. Ємець – Полтава : РВВ ПУЕТ, 2017. – 69с.
2. Панішев А.В. Вступ до теорії складності дискретних задач : Монографія. / А.В. Панішев – Ж. : ЖДТУ, 2004. – 236с.
3. Ємець О. О. Дистанційний курс ПУЕТ «Алгоритми та структури даних» для студентів спеціальностей «Комп'ютерні науки та інформаційні технології» / О. О. Ємець. – [Електронний ресурс].
4. Алгоритм сортування [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії. Режим доступу:
https://uk.wikipedia.org/wiki/Алгоритм_сортування
5. Алгоритми сортування порівняння [Електронний ресурс]. Режим доступу:
<http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html>
6. Сортування включенням [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії. Режим доступу:
https://uk.wikipedia.org/wiki/Сортування_включенням
7. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.
8. Шульга І. І. Тренажер «Рекурсивне породження переставлень» / І. І. Шульга, О. О. Ємець // Комп'ютерні науки і прикладна математика (КНіПМ-2020): матеріали наук.-практ. семінару. Випуск 5. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2020. – С.12-16. – Режим доступу: <http://dspace.puet.edu.ua/handle/123456789/8270>.

9. C Sharp [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії.

Режим доступу:

https://ru.wikipedia.org/wiki/C_Sharp

10. Microsoft Visual Studio [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії. Режим доступу:

https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio